KeyStone I Architecture
Antenna Interface 2 (AIF2)

User's Guide

Literature Number: SPRUGV7E
November 2010–Revised February 2015

Contents

Preface **28**

1 Introduction **30**

1.1 Purpose 31

1.2 Scope 31

1.3 Terminology 31

1.3.1 Abbreviations and Acronyms 31

1.3.2 Definitions 33

1.4 Features 33

2 Overview of AIF2 Hardware and Software Components **37**

3 Radio Standard Requirements and AIF2 Handling **40**

3.1 WCDMA Requirements and Handling 41

3.2 OFDM Requirements and Handling (LTE FDD, LTE TDD, WiMax, TD-SCDMA) 41

3.2.1 OFDM DMA 41

3.2.1.1 OFDM TDD DMA {LTE TDD, WiMax, TD-SCDMA} 43

3.2.2 LTE 43

3.2.2.1 LTE Framing 44

3.2.2.2 CPRI LTE (FDD and TDD) 46

3.2.2.3 OBSAI LTE (FDD and TDD) 47

3.2.3 LTE 80 MHz and 100MHz Support 48

3.2.3.1 80 MHz 48

3.2.3.2 100 MHz 48

3.2.4 TD-SCDMA 48

3.2.4.1 TD-SCDMA Framing 48

3.2.4.2 CPRI TD-SCDMA 50

3.2.4.3 TD-SCDMA DMA 51

3.2.4.4 TD-SCMDA Timer (AT) Operation 52

3.2.5 WiMax 52

3.2.5.1 WiMax Framing 52

3.2.5.2 WiMax, AIF2 Implementation 53

3.2.5.3 OBSAI WiMax 53

3.2.5.4 CPRI WiMax 54

3.2.6 GSM/Edge Requirement and Handling 54

3.2.6.1 GSM Base Band Hopping 55

3.2.6.2 AxC Compression (Time Slot DTX) 56

3.2.6.3 Dynamic Configuration 56

4 AIF2 Timing Information **60**

4.1 Timing and Topology 61

4.1.1 Synchronization and Resynchronization 63

4.2 AxC Offset 63

4.3 OBSAI Reception Timing 65

4.3.1 Pi and Delta Timing 65

4.3.2 OBSAI Radio Timing 66

4.3.3 Tx Timing Message Construction: Radio Counter 66

4.3.4 AIF2 Implementation of OBSAI Radio Timing 67

4.3.4.1 AIF2 OBSAI Ingress Reception Timing Implementation 67
4.3.4.2 AIF2 OBSAI Egress Transmission Timing Implementation 68
4.4 PhyT and RadT Timers 70
4.5 AIF2 Timing Details 71
4.5.1 DMA Timing 71
4.5.1.1 PKTDMA Timing 71
4.5.1.2 AD DIO DMA Timing 73
4.5.2 AIF2 Internal Delay 73
4.5.2.1 DB =>PE Timing 73
4.5.2.2 PE=>TM Timing 74
4.5.2.3 Delta and PE Event Offset Calculation 74
4.5.2.4 TM =>RM SerDes Loopback Timing 74
4.5.2.5 TM =>RM Timing 74
4.5.2.6 RM => RT timing (Retransmission or Aggregation) 75
4.5.2.7 RM =>PD =>DB Timing (UL Case) 75
4.5.2.8 MAC Timing (SerDes Timing) 76
4.5.3 WCDMA Timing Example 77
4.5.4 LTE Timing Example 78
4.5.5 CLKSTP_REQ Min Timing 79
5 Interface Standards CPRI, OBSAI Specifics 81
5.1 CPRI/OBSAI Overlay of Functionality 82
5.1.1 CPRI/OBSAI Ingress DMA Channel Addressing/Indexing 82
5.1.1.1 CPRI 82
5.1.1.2 OBSAI 83
5.1.2 Reception Timing 83
5.1.3 Transmission Based Timing 83
5.1.4 Short Frame Mode 84
5.1.4.1 OBSAI 84
5.1.4.2 CPRI 84
5.2 CPRI Specifics 85
5.2.1 CPRI Control Data Flow 85
5.2.1.1 Fast Ethernet 85
5.2.1.2 4B/5B Encoding 86
5.2.1.3 Interleave/Deinterleave 88
5.2.1.4 Packet Parsing 88
5.2.1.5 Control Packet DMA 89
5.2.2 CPRI Physical Interface (Phy) Timing 89
5.2.3 CPRI Link Maintenance {LOS, LOF, RAI, SDI} 89
5.2.4 CPRI AIF2 Timer Startup 90
5.3 OBSAI Specifics 91
5.3.1 OBSAI Standard Overview 91
5.3.1.1 OBSAI Overview: Frame/Message Structure 93
5.3.1.2 OBSAI Overview: RP3 Tx and Rx FSMs 94
5.3.1.3 OBSAI Overview: Timestamp 96
5.3.1.4 OBSAI Overview: Message Payload Formats 97
5.3.2 OBSAI: Header Based Processing 98
5.3.2.1 DMA Channel Index 98
5.3.2.2 8B/10B Code Violation and Bad OBSAI Message Header 98
5.3.2.3 OBSAI: Missing Timestamp 99
5.3.3 OBSAI: 6 GHz Scrambling 100
5.3.4 OBSAI: Generic Packet Mode 101
5.3.5 OBSAI: RP3 Transmission 102
5.3.5.1 OBSAI Tx: Modulo Rules 102

5.3.5.2 OBSAI Tx: Dual-Bit Map Rules 102
6 AIF2 DMA (Multicore Navigator and DIO) Information 104
6.1 VBUS Data Formats 105
6.1.1 AIF2 16-Bit Sample DMA (OFDM and WCDMA DL) 105
6.1.2 WCDMA UL DMA 106
6.1.3 Generic Format DMA 107
6.1.4 VBUS Config Bus 108
6.2 AIF2 Direct IO 108
6.2.1 AIF2 Direct IO 109
6.2.2 AIF2 Direct IO Link Data Trace 110
6.3 PKTDMA 111
6.3.1 Multicore Navigator Buffers and Descriptors 112
6.3.2 Multicore Navigator Queues 113
6.3.3 Multicore Navigator and DIO Scheduling 113
6.3.4 Multicore Navigator Packet Types 114
6.4 Multicore Navigator Examples 117
6.4.1 Ingress (AIF2 Reception DMA-to-L2 RAM) 117
6.4.2 Egress (L2 RAM DMA-to-AIF2 Transmission) 119
7 Implementation Details and Application for Internal Modules 121
7.1 AIF2 Architecture 122
7.2 VC (VBUSP Configuration Bus Interface) 123
7.2.1 Clock Stop and Emulation 123
7.2.2 Emulation Control 123
7.2.3 AIF2 Reset 124
7.3 SD (SerDes) 125
7.3.1 AIF2 Clock Strategy 125
7.3.2 SerDes Serial Bit Ordering 126
7.3.3 SerDes Common Configuration 127
7.3.3.1 MPY Setup 127
7.3.3.2 VCO Speed Range 127
7.3.3.3 refclkp/n Jitter and PLL Loop Bandwidth 127
7.3.3.4 Clock Bypass 128
7.3.4 SerDes Rx Configuration 128
7.3.4.1 Data Rate 128
7.3.4.2 Symbol Alignment 128
7.3.4.3 Clock Recovery Algorithms 128
7.3.4.4 Rx Termination 129
7.3.4.5 Equalization, EQ Hold, and ENOC 129
7.3.4.6 Test Pattern and Loopback Setting 129
7.3.5 SerDes Tx Configuration 129
7.3.5.1 Data Rate 129
7.3.5.2 MSYNC 129
7.3.5.3 Swing, TWPRE and TWPST 130
7.3.5.4 FIRUPT 131
7.3.5.5 Test Pattern and Loopback Setting 131
7.4 RM (RX Mac) 131
7.4.1 Receive Clock Synchronization 131
7.4.1.1 Clock Detection Function 131
7.4.2 Receive Frame Synchronization 133
7.4.2.1 Receive Frame Synchronizer State Machine 133
7.4.3 Receive Frame Formatting 136
7.4.3.1 Line Code Violation Handling 136
7.4.3.2 Received K character Handling 136

7.4.4 RM Programming 137
7.4.4.1 RM Link Configuration 137
7.4.4.2 RM FSM Sync Count Configuration 138
7.4.4.3 RM FSM Unsync Count Configuration 138
7.4.4.4 Clock Detection Configuration 138
7.4.4.5 RM Status 138
7.5 TM (TX Mac) 139
7.5.1 Transmit Frame Synchronization and Formatting 139
7.5.1.1 Frame Construction 140
7.5.1.2 Transmit Frame Control 140
7.5.2 Transmit FIFO 141
7.5.3 Tx Mac Programming 141
7.5.3.1 TM State Machine Control Register (AIF2_TM_CTRL[0-5]) 141
7.5.3.2 TM Scrambler Control Register (AIF2_TM_SCR_CTRL[0-5]) 142
7.5.3.3 TM CPRI L1 Inband Configuration Register (AIF2_TM_L1_CFG[0-5]) 142
7.5.3.4 TM CPRI L1 Inband Enable Register (AIF2_TM_L1_EN[0-5]) 142
7.5.3.5 TM CPRI L1 Inband LOSERR Link Select Register (AIF2_TM_LOSERR[0-5]) 143
7.5.3.6 TM CPRI L1 Inband LOFERR Link Select Register (AIF2_TM_LOFERR[0-5]) 143
7.5.3.7 TM CPRI L1 Inband LOSRx Link Select Register (AIF2_TM_LOSRx[0-5]) 143
7.5.3.8 TM CPRI L1 Inband LOFRx Link Select Register (AIF2_TM_LOFRx[0-5]) 143
7.5.3.9 TM CPRI L1 Inband RAIRx Link Select Register (AIF2_TM_RAIRx[0-5]) 144
7.5.3.10 TM CPRI Pointer P Configuration Register (AIF2_TM_PTRP[0-5]) 144
7.5.3.11 TM CPRI Startup Configuration Register (AIF2_TM_STRT[0-5]) 144
7.5.3.12 TM CPRI Protocol Version Configuration (AIF2_TM_PROT[0-5]) 144
7.6 CI and CO (CPRI Input and Output Data Module) 144
7.6.1 CPRI Data Formats 144
7.7 RT (Retransmitter) 150
7.7.1 15-bit Saturation Operation for CPRI (7-bit follows the same rule) 151
7.7.2 16-bit Saturation Operation (8-bit follows the same rule) 151
7.8 PD (Protocol Decoder) 151
7.8.1 PD Route Module 151
7.8.1.1 OBSAI Route and DB buffer Channel Number 151
7.8.1.2 CPRI Route (DBM) and DB Buffer Channel Number 152
7.8.2 PD Pack Module 153
7.8.2.1 PD Pack CPRI Packet Delimitation 153
7.8.2.2 PD Pack CPRI Null Delimitation 154
7.8.2.3 PD Pack OBSAI Packet Delimitation 155
7.8.2.4 PD Pack 4 Channel Map 156
7.8.2.5 PD Pack Ethernet Header Strip 156
7.8.2.6 PD CRC 157
7.8.3 PD Stage Module 158
7.8.3.1 PD Stage 158
7.8.3.2 PD FRAME 158
7.8.3.3 PD DMA_IF 162
7.9 PE (Protocol Encoder) 162
7.9.1 OBSAI Transmission Rules 162
7.9.1.1 PE OBSAI Modulo Transmission Rules 163
7.9.1.2 PE OBSAI Dual Bit Map Tx Rules 164
7.9.1.3 PE Transmission Rules Channel LUT 165
7.9.2 CPRI Transmission Rule 166
7.9.3 PE Pack 167
7.9.4 PE Timer 167
7.9.5 PE Frame 168

7.9.5.1 AxC Frame Counter 168
7.9.5.2 Channel and Packet State 169
7.9.6 PE DB Interface 169
7.9.6.1 CPRI Data Request I/F 169
7.9.6.2 OBSAI Data Request I/F 170
7.9.6.3 DIO Data Request I/F 170
7.9.6.4 PE DB Token Request I/F 170
7.9.7 PE Message Construction 171
7.9.7.1 PE_MC_ENET 171
7.9.7.2 PE_MC_CRC 171
7.9.7.3 PE_MC_OBSAI_HEADER 172
7.9.8 PE RT Interface 172
7.9.8.1 PE_RT_SYNC 172
7.10 DB (Data Buffer) 172
7.10.1 Addressing 173
7.10.2 Alignment Logic 174
7.10.3 Buffer Channel Enables 175
7.10.4 Other Functionality 175
7.10.4.1 Protocol-Specific Data 175
7.10.4.2 Packet Type 175
7.10.4.3 Egress DIO offset 175
7.10.4.4 EDB Packet Mode Control 175
7.10.4.5 Consumption-Based Scheduling 176
7.10.5 DB Debug 176
7.10.5.1 Overview 176
7.10.5.2 Ingress DB Debug 176
7.10.5.3 Ingress DB Activity Monitoring 178
7.10.5.4 Egress DB Debug 178
7.10.5.5 Egress DB Activity Monitoring 179
7.10.6 DB Data Trace Feature 180
7.11 AD (AIF2 DMA) 182
7.11.1 Ingress Scheduler (ISCH) 182
7.11.2 Egress Scheduler (ESCH) 182
7.11.3 Direct IO 183
7.11.3.1 RAC Example 185
7.11.3.2 L2 Example 186
7.11.3.3 DDR3 Example 186
7.11.3.4 TAC Example 188
7.11.3.5 Configuration Registers 188
7.11.3.6 Data Trace DIO 189
7.12 AT (AIF2 Timer) 190
7.12.1 AT Phy Timer (PHYT RP3 Timer) 192
7.12.2 AT Radio Timer (RADT RP1 Timer) 194
7.12.3 AT OBSAI RP1 Synchronization 196
7.12.4 AT Event Generation 199
7.12.5 AT WCDMA Counter 201
7.12.6 AT LTE Counter 202
7.12.7 AT WiMAX Counter 203
7.12.8 AT TD-SCDMA 204
7.12.9 AT GSM Counter 205
7.13 EE (Error and Exception Handler) 208
7.13.1 EE Interrupt Support 209
7.13.2 Mapping Error/Alarm Conditions to Interrupts 211

7.13.2.1 Non-PKTDMA Error/Alarm Condition Mapping 211
7.13.2.2 PKTDMA Error/Alarm Condition Mapping 212
7.13.3 End-of-Interrupt Support (EOI) 212
7.13.4 EE Run (EE_AIF2_RUN) 213
7.13.5 Error/Alarm Condition Force Support 213
7.13.6 Error/Alarm Condition Origination Support 213
8 Register Map 216
8.1 VC Registers 217
8.1.1 Grouped Common Registers Details 217
8.1.1.1 aif2_pid Register (Address = 0x0000) 217
8.1.1.2 aif2_scratch Register (Address = 0x0004) 218
8.1.1.3 aif2_reset Register (Address = 0x0008) 219
8.1.1.4 aif2_emu Register (Address = 0x000C) 220
8.1.1.5 vc_stat Register (Address = 0x0010) 221
8.2 SD Registers 222
8.2.1 Grouped Common Registers Details 224
8.2.1.1 SD_RX_EN_CFG[0] Register (offset = 0x8000) 224
8.2.1.2 SD_RX_R1_CFG[0] Register (offset = 0x8004) 225
8.2.1.3 SD_RX_R2_CFG[0] Register (offset = 0x8008) 226
8.2.1.4 SD_RX_STS[0] Register (offset = 0x800C) 227
8.2.1.5 SD_TX_EN_CFG[0] Register (offset = 0x8010) 228
8.2.1.6 SD_TX_R1_CFG[0] Register (offset = 0x8014) 229
8.2.1.7 SD_TX_R2_CFG[0] Register (offset = 0x8018) 230
8.2.1.8 SD_TX_STS[0] Register (offset = 0x801C) 232
8.2.1.9 SD_PLL_B8_EN_CFG Register (offset = 0xB000) 233
8.2.1.10 SD_PLL_B4_EN_CFG Register (offset = 0xB004) 234
8.2.1.11 SD_PLL_B8_CFG Register (offset = 0xB008) 235
8.2.1.12 SD_PLL_B4_CFG Register (offset = 0xB00C) 236
8.2.1.13 SD_PLL_B8_STS Register (offset = 0xB010) 237
8.2.1.14 SD_PLL_B4_STS Register (offset = 0xB014) 238
8.2.1.15 SD_CLK_SEL_CFG Register (offset = 0xB018) 239
8.2.1.16 SD_LK_CLK_DIS_CFG Register (offset = 0xB01C) 240
8.3 RM Registers 241
8.3.1 Link Registers Details 243
8.3.1.1 RM_LK_CFG0[0] Register (offset = 0x50000) 243
8.3.1.2 RM_LK_CFG1[0] Register (offset = 0x50004) 244
8.3.1.3 RM_LK_CFG2[0] Register (offset = 0x50008) 245
8.3.1.4 RM_LK_CFG3[0] Register (offset = 0x5000C) 246
8.3.1.5 RM_LK_CFG4[0] Register (offset = 0x50010) 247
8.3.1.6 RM_LK_STS0[0] Register (offset = 0x50014) 248
8.3.1.7 RM_LK_STS1[0] Register (offset = 0x50018) 249
8.3.1.8 RM_LK_STS2[0] Register (offset = 0x5001C) 250
8.3.1.9 RM_LK_STS3[0] Register (offset = 0x50020) 251
8.3.1.10 RM_LK_STS4[0] Register (offset = 0x50024) 252
8.3.1.11 RM_CFG Register (offset = 0x53000) 253
8.4 TM Registers 254
8.4.1 Link Registers Details 258
8.4.1.1 TM_LK_CFG[0] Register (offset = 0x4C000) 258
8.4.1.2 TM_LK_CTRL[0] Register (offset = 0x4C004) 259
8.4.1.3 TM_LK_SCR_CTRL[0] Register (offset = 0x4C008) 260
8.4.1.4 TM_LK_L1_CFG[0] Register (offset = 0x4C00C) 261
8.4.1.5 TM_LK_L1_EN[0] Register (offset = 0x4C010) 262
8.4.1.6 TM_LK_LOSERR[0] Register (offset = 0x4C014) 263

8.4.1.7 TM_LK_LOFERR[0] Register (offset = 0x4C018) 264
8.4.1.8 TM_LK_LOSRx[0] Register (offset = 0x4C01C) 265
8.4.1.9 TM_LK_LOFRx[0] Register (offset = 0x4C020) 266
8.4.1.10 TM_LK_RAIRx[0] Register (offset = 0x4C024) 267
8.4.1.11 TM_LK_HFN[0] Register (offset = 0x4C028) 268
8.4.1.12 TM_LK_PTRP[0] Register (offset = 0x4C02C) 269
8.4.1.13 TM_LK_STRT[0] Register (offset = 0x4C030) 270
8.4.1.14 TM_LK_PROT[0] Register (offset = 0x4C034) 271
8.4.1.15 TM_LK_STAT[0] Register (offset = 0x4C038) 272
8.4.1.16 TM_FRM_MODE[0] Register (offset = 0x4C03C) 273
8.5 CI Registers 274
8.5.1 Link Registers Details 274
8.5.1.1 CI_LK_CFG[0] Register (offset = 0x58000) 274
8.6 CO Registers 275
8.6.1 Link Registers Details 275
8.6.1.1 CO_LK_CFG[0] Register (offset = 0x5C000) 275
8.7 RT Registers 276
8.7.1 Link Registers Details 277
8.7.1.1 RT_LK_CFG[0] Register (offset = 0x54000) 277
8.7.1.2 RT_LK_DPTH[0] Register (offset = 0x54004) 278
8.7.1.3 RT_HDR_ERR[0] Register (offset = 0x54008) 279
8.7.1.4 RT_LK_STAT[0] Register (offset = 0x5400C) 280
8.8 PD Registers 281
8.8.1 Link Registers Details 285
8.8.1.1 pd_link_a[0] Register (offset = 0x60000) 285
8.8.1.2 pd_link_b[0] Register (offset = 0x60004) 286
8.8.1.3 pd_lk_pack_cpri[0] Register (offset = 0x60008) 287
8.8.1.4 pd_cpri_crc[0] Register (offset = 0x60014) 288
8.8.1.5 pd_pack_map[0] Register (offset = 0x60018) 289
8.8.1.6 pd_dbm[0] Register (offset = 0x6001C) 290
8.8.1.7 pd_dbm_1map[0][0-3] Register (offset = 0x60020) 291
8.8.1.8 pd_dbm_2map[0][0-2] Register (offset = 0x60030) 292
8.8.1.9 pd_type_lut[0][0-31] Register (offset = 0x60080) 293
8.8.1.10 pd_cpri_id_lut[0][0-127] Register (offset = 0x60200) 294
8.8.1.11 pd_cw_lut[0][0-255] Register (offset = 0x60400) 295
8.8.1.12 pd_global Register (offset = 063000) 296
8.8.1.13 pd_global_en_set Register (0x63004) 297
8.8.1.14 pd_global_en_clr Register (offset = 0x63008) 298
8.8.1.15 pd_global_en_sts Register (offset = 0x6300C) 299
8.8.1.16 pd_dma Register (offset = 0x63010) 300
8.8.1.17 pd_radt_tc Register (offset = 0x63014) 301
8.8.1.18 pd_chan_sts[4] Register (offset = 0x63020) 302
8.8.1.19 pd_pkt_sts[4] Register (offset = 0x63030) 303
8.8.1.20 pd_frm_tc[6] Register (offset = 0x63040) 304
8.8.1.21 pd_route[128] Register (offset = 0x63400) 305
8.8.1.22 pd_dmachan[128] Register (offset = 0x63600) 306
8.8.1.23 pd_dmachan_a[128] Register (offset = 0x64000) 307
8.8.1.24 pd_dmachan_b[128] Register (offset = 0x64200) 308
8.8.1.25 pd_dmachan_c[128] Register (offset = 0x64400) 309
8.8.1.26 pd_dmachan_d[128] Register (offset = 0x64600) 310
8.8.1.27 pd_dmachan_e[128] Register (offset = 0x64800) 311
8.8.1.28 pd_dmachan_f[128] Register (offset = 0x64A00) 312
8.8.1.29 pd_frm_msg_tc[128] Register (offset = 64C00) 313

8.9 PE Registers 314
8.9.1 Link Registers Details 317
8.9.1.1 pe_link[0] Register (offset = 0x68000) 317
8.9.1.2 pe_crc[0] Register (offset = 0x68004) 318
8.9.1.3 pe_cpri_dbm[0] Register (offset = 0x68008) 319
8.9.1.4 pe_cpri_bitswap[0] Register (offset = 0x6800C) 320
8.9.1.5 pe_cpridbm_1map[0][0-3] Register (offset = 0x68010) 321
8.9.1.6 pe_cpridbm_2map[0][0-2] Register (offset = 0x68020) 322
8.9.1.7 pe_cpri0[0] Register (offset = 0x6802C) 323
8.9.1.8 pe_cpri1[0] Register (offset = 0x68030) 324
8.9.1.9 pe_cw_lut[0][0-255] Register (offset = 0x68400) 325
8.9.1.10 pe_global Register (offset = 0x6C000) 326
8.9.1.11 pe_global_en_set Register (offset = 0x6C004) 327
8.9.1.12 pe_global_en_clr (offset = 0x6C008) 327
8.9.1.13 pe_global_en_sts Register (offset = 0x6C00C) 328
8.9.1.14 pe_chan_sts[4] Register (offset = 0x6C010) 329
8.9.1.15 pe_pkt_sts[4] Register (offset = 0x6C020) 330
8.9.1.16 pe_frm_tc[6] Register (offset = 0x6C080) 331
8.9.1.17 pe_dmachan_en[128] Register (offset = 0x6C200) 332
8.9.1.18 pe_dma0chan[128] Register (offset = 0x6C400) 333
8.9.1.19 pe_in_fifo[128] Register (offset = 0x6C600) 334
8.9.1.20 pe_axc_offset[128] Register (offset = 0x6C800) 335
8.9.1.21 pe_frm_msg_tc[128] Register (offset = 0x6CA00) 336
8.9.1.22 pe_modtxrule[64] Register (offset = 0x6CC00) 337
8.9.1.23 pe_obsai_hdr[0-127] Register (offset = 0x6CE00) 338
8.9.1.24 pe_obsai_dbm[64] Register (offset = 0x70000) 339
8.9.1.25 pe_dbm_map[512] Register (offset = 0x70800) 340
8.9.1.26 pe_rule_chanlut0[512] Register (offset = 0x74000) 341
8.9.1.27 pe_rule_chanlut1[512] Register (offset = 0x74800) 342
8.9.1.28 pe_rule_chanlut2[512] Register (offset = 0x75000) 343
8.9.1.29 pe_rule_chanlut3[512] Register (offset = 0x75800) 344
8.9.1.30 pe_rule_chanlut4[512] Register (offset = 0x76000) 345
8.9.1.31 pe_rule_chanlut5[512] Register (offset = 0x76800) 346
8.9.1.32 pe_rule_chanlut6[512] Register (offset = 0x77000) 347
8.9.1.33 pe_rule_chanlut7[512] Register (offset = 0x77800) 348
8.10 DB Registers 349
8.10.1 Grouped Common Registers Details 351
8.10.1.1 DB_IDB_CFG Register (offset = 0x10000) 351
8.10.1.2 DB_IDB_GLOBAL_EN_SET Register (offset = 0x10004) 352
8.10.1.3 DB_IDB_GLOBAL_EN_CLR Register (offset = 0x10008) 353
8.10.1.4 DB_IDB_GLOBAL_EN_STS Register (offset = 0x1000C) 354
8.10.1.5 DB_IDB_CH_EN[4] Register (offset = 0x10010) 355
8.10.1.6 DB_IDB_DEBUG_D0 Register (offset = 0x10100) 356
8.10.1.7 DB_IDB_DEBUG_D1 Register (offset = 0x10104) 357
8.10.1.8 DB_IDB_DEBUG_D2 Register (offset = 0x10108) 358
8.10.1.9 DB_IDB_DEBUG_D3 Register (offset = 0x1010C) 359
8.10.1.10 DB_IDB_DEBUG_SBND Register (offset = 0x10110) 360
8.10.1.11 DB_IDB_DEBUG_DB_WR Register (offset = 0x10114) 361
8.10.1.12 DB_IDB_DEBUG_OFS Register (offset = 0x10118) 362
8.10.1.13 DB_IDB_DEBUG_OFS_DAT Register (offset = 0x1011C) 363
8.10.1.14 DB_IDB_PTR_CH[128] Register (offset = 0x10200) 364
8.10.1.15 DB_IDB_CFG_CH[128] Register (offset = 0x10400) 365
8.10.1.16 DB_IDB_CH_EMPTY[4] Register (offset = 0x10600) 366

8.10.1.17 DB_EDB_CFG Register (offset = 0x11000) 367
8.10.1.18 DB_EDB_GLOBAL_EN_SET Register (offset = 0x11004) 368
8.10.1.19 DB_EDB_GLOBAL_EN_CLR Register (offset = 0x11008) 369
8.10.1.20 DB_EDB_GLOBAL_EN_STS Register (offset = 0x1100C) 370
8.10.1.21 DB_EDB_CH_EN[4] Register (offset = 0x11010) 371
8.10.1.22 DB_EDB_DEBUG_D0 Register (offset = 0x11100) 372
8.10.1.23 DB_EDB_DEBUG_D1 Register (offset = 0x11104) 373
8.10.1.24 DB_EDB_DEBUG_D2 Register (offset = 0x11108) 374
8.10.1.25 DB_EDB_DEBUG_D3 Register (offset = 0x1110C) 375
8.10.1.26 DB_EDB_DEBUG_SBND Register (offset = 0x11110) 376
8.10.1.27 DB_EDB_DEBUG_RD_CNTL Register (offset = 0x11114) 377
8.10.1.28 DB_EDB_DEBUG_DB_RD Register (offset = 0x1111C) 378
8.10.1.29 DB_EDB_DEBUG_OFS Register (offset = 0x11120) 379
8.10.1.30 DB_EDB_DEBUG_OFS_DAT Register (offset = 0x11124) 380
8.10.1.31 DB_EDB_DEBUG_WR_TOK Register (offset = 0x11128) 381
8.10.1.32 DB_EDB_EOP_CNT Register (offset = 0x1112C) 382
8.10.1.33 DB_EDB_PTR_CH[128] Register (offset = 0x11200) 383
8.10.1.34 DB_EDB_CFG_CH[128] Register (offset = 0x11400) 384
8.11 AD Registers 385
8.11.1 Grouped Common Registers Details 388
8.11.1.1 AD_DIO_I_TABLE_SEL[0] Register (offset = 0xC000) 388
8.11.1.2 AD_DIO_I_TABLE_LOOP_CFG[0] Register (offset = 0xC004) 389
8.11.1.3 AD_DIO_I_DMA_CFG0[0] Register (offset = 0xC008) 390
8.11.1.4 AD_DIO_I_DMA_CFG1[0] Register (offset = 0xC00C) 391
8.11.1.5 AD_DIO_I_DMA_CFG2[0] Register (offset = 0xC010) 392
8.11.1.6 AD_DIO_I_BCN_TABLE0_ROW0[0] to AD_DIO_I_BCN_TABLE0_ROW15[0] Register (offset = 0xC014 to 0xC050) 393
8.11.1.7 AD_DIO_I_BCN_TABLE1_ROW0[0] to AD_DIO_I_BCN_TABLE1_ROW15[0] Register (offset = 0xC054 to 0xC090) 394
8.11.1.8 AD_DIO_E_TABLE_SEL[0] Register (offset = 0xC300) 395
8.11.1.9 AD_DIO_E_TABLE_LOOP_CFG[0] Register (offset = 0xC304) 396
8.11.1.10 AD_DIO_E_DMA_CFG0[0] Register (offset = 0xC308) 397
8.11.1.11 AD_DIO_E_DMA_CFG1[0] Register (offset = 0xC30C) 398
8.11.1.12 AD_DIO_E_DMA_CFG2[0] Register (offset = 0xC310) 399
8.11.1.13 AD_DIO_E_BCN_TABLE0_ROW0[0] to AD_DIO_E_BCN_TABLE0_ROW15[0] Register (offset = 0xC314 to 0xC350) 400
8.11.1.14 AD_DIO_E_BCN_TABLE1_ROW0[0] to AD_DIO_E_BCN_TABLE1_ROW15[0] Register (offset = 0xC354 to 0xC390) 401
8.11.1.15 AD_DIO_DT_DMA_CFG0 Register (offset = 0xC600) 402
8.11.1.16 AD_DIO_DT_DMA_CFG1 Register (offset = 0xC604) 403
8.11.1.17 AD_DIO_DT_DMA_CFG2 Register (offset = 0xC608) 404
8.11.1.18 AD_DIO_DT_DMA_CFG3n Register (offset = 0xC60C) 405
8.11.1.19 AD_DIO_I_GLOBAL_EN_SET Register (offset = 0xC610) 406
8.11.1.20 AD_ DIO_I _GLOBAL_EN_CLR Register (offset = 0xC614) 407
8.11.1.21 AD_ DIO_I _GLOBAL_EN_STS Register (offset = 0xC618) 408
8.11.1.22 AD_DIO_E_GLOBAL_EN_SET Register (offset = 0xC61C) 409
8.11.1.23 AD_ DIO_E _GLOBAL_EN_CLR Register (offset = 0xC620) 410
8.11.1.24 AD_ DIO_E_GLOBAL_EN_STS Register (offset = 0xC624) 411
8.11.1.25 AD_ISCH_CFG Register (offset = 0xE000) 412
8.11.1.26 AD_ISCH_GLOBAL_EN_SET Register (offset = 0xE004) 413
8.11.1.27 AD_ISCH_GLOBAL_EN_CLR Register (offset = 0xE008) 414
8.11.1.28 AD_ISCH_GLOBAL_EN_STS Register (offset = 0xE00C) 415
8.11.1.29 AD_ISCH_EOP_CNT Register (offset = 0xE010) 416
8.11.1.30 AD_ESCH_CFG Register (offset = 0xE100) 417

8.11.1.31 AD_ESCH_GLOBAL_EN_SET Register (offset = 0xE104) 418
8.11.1.32 AD_ESCH_GLOBAL_EN_CLR Register (offset = 0xE108) 419
8.11.1.33 AD_ESCH_GLOBAL_EN_STS Register (offset = 0xE10C) 420
8.12 AT Registers 421
8.12.1 Grouped Common Registers Details 427
8.12.1.1 at_pimax_lk[0] Register (offset = 0x48100) 427
8.12.1.2 at_pimin_lk[0] Register (offset = 0x48104) 428
8.12.1.3 at_pivalue_lk[0] Register (offset = 0x48108) 429
8.12.1.4 at_event_offset[0] Register (offset = 0x48200) 430
8.12.1.5 at_event_mod_tc[0] Register (offset = 0x48204) 431
8.12.1.6 at_event_mask_lsbs[0] Register (offset = 0x48204) 432
8.12.1.7 at_event_mask_msbs[0] Register (offset = 0x4820C) 433
8.12.1.8 at_ad_ingr_event_offset[0] Register (offset = 0x48400) 434
8.12.1.9 at_ad_ingr_event_mod_tc[0] Register (offset = 0x48404) 435
8.12.1.10 at_ad_egr_event_offset[0] Register (offset = 0x48500) 436
8.12.1.11 at_ad_egr_event_mod_tc[0] Register (offset = 0x48504) 437
8.12.1.12 at_tm_delta_event_offset[0] Register (offset = 0x48540) 438
8.12.1.13 at_tm_delta_event_mod_tc[0] Register (offset = 0x48544) 439
8.12.1.14 at_pe_event_offset[0] Register (offset = 0x48580) 440
8.12.1.15 at_pe_event_mod_tc[0] Register (offset = 0x48584) 441
8.12.1.16 at_pe_event2_offset[0] Register (offset = 0x485B0) 442
8.12.1.17 at_pe_event2_mod_tc[0] Register (offset = 0x485B4) 443
8.12.1.18 at_control1 Register (offset = 0x48000) 444
8.12.1.19 at_control2 Register (offset = 0x48004) 445
8.12.1.20 at_sw_sync Register (offset = 0x48008) 446
8.12.1.21 at_phyt_cmp_radsync Register (ofset = 0x4800C) 447
8.12.1.22 at_rp1_type Register (offset = 0x48010) 448
8.12.1.23 at_captradt Register (offset = 0x48014) 449
8.12.1.24 at_rp1_type_capture Register (offset = 0x48020) 450
8.12.1.25 at_rp1_tod_capture_l Register (offset = 0x48024) 451
8.12.1.26 at_rp1_tod_capture_h Register (offset = 0x48028) 452
8.12.1.27 at_rp1_rp3_capture_l Register (offset = 0x4802C) 453
8.12.1.28 at_rp1_rp3_capture_h Register (offset = 0x48030) 454
8.12.1.29 at_rp1_rad_capture_l Register (offset = 0x48034) 455
8.12.1.30 at_rp1_rad_capture_h Register (offset = 0x48038) 456
8.12.1.31 at_phyt_clkcnt_value Register (offset = 0x48040) 457
8.12.1.32 at_phyt_frm_value_lsbs Register (offset = 0x48044) 458
8.12.1.33 at_phyt_frm_value_msbs Register (offset = 0x48048) 459
8.12.1.34 at_radt_value_lsbs Register (offset = 0x4804C) 460
8.12.1.35 t_radt_value_mid Register (offset = 0x48050) 461
8.12.1.36 at_radt_value_msbs Register (offset = 0x48054) 462
8.12.1.37 at_ulradt_value_lsbs Register (offset = 0x48058) 463
8.12.1.38 at_ulradt_value_mid Register (offset = 0x4805C) 464
8.12.1.39 at_ulradt_value_msbs Register (offset = 0x48060) 465
8.12.1.40 at_dlradt_value_lsbs Register (offset = 0x48064) 466
8.12.1.41 at_dlradt_value_mid Register (offset = 0x48068) 467
8.12.1.42 at_dlradt_value_msbs Register (offset = 0x4806C) 468
8.12.1.43 at_radt_wcdma_value Register (offset = 0x48070) 469
8.12.1.44 at_ulradt_wcdma_value Register (offset = 0x48074) 470
8.12.1.45 at_dlradt_wcdma_value Register (offset = 0x48078) 471
8.12.1.46 at_radt_wcdma_div Register (offset = 0x4807C) 472
8.12.1.47 at_phyt_init_lsbs Register (offset = 0x48080) 473
8.12.1.48 at_phyt_init_mid Register (offset = 0x48084) 474

8.12.1.49 at_phyt_init_msbs Register (offset = 0x48088) 475
8.12.1.50 at_phyt_tc_lsbs Register (offset = 0x4808C) 476
8.12.1.51 at_phyt_frame_tc_lsbs Register (offset = 0x48090) 477
8.12.1.52 at_phyt_frame_tc_msbs Register (offset = 0x48094) 478
8.12.1.53 at_radt_init_lsbs Register (offset = 048098) 479
8.12.1.54 at_radt_init_mid Register (offset = 0x4809C) 480
8.12.1.55 at_radt_init_msbs Register (offset = 0x480A0) 481
8.12.1.56 at_ulradt_init_lsbs Register (offset = 0x480A4) 482
8.12.1.57 at_radt_tstamp_value Register (offset = 0x480A8) 483
8.12.1.58 at_dlradt_init_lsbs Register (offset = 0x480B0) 484
8.12.1.59 at_gsm_tcount_init Register (offset = 0x480B4) 485
8.12.1.60 at_gsm_tcount_value Register (offset = 0x480D8) 486
8.12.1.61 at_radt_symb_lut_index_tc Register (offset = 0x480BC) 487
8.12.1.62 at_radt_frame_tc_lsbs Register (offset = 0x480C8) 488
8.12.1.63 at_radt_frame_tc_msbs Register (offset = 0x480CC) 489
8.12.1.64 at_ulradt_init_mid Register (offset = 0x480E0) 490
8.12.1.65 at_ulradt_init_msbs Register (offset = 0x480E4) 491
8.12.1.66 at_dlradt_init_mid Register (offset = 0x480E8) 492
8.12.1.67 at_dlradt_init_msbs Register (offset = 0x480EC) 493
8.12.1.68 at_radt_init_lut_index Register (offset = 0x480F4) 494
8.12.1.69 at_ulradt_init_lut_index Register (offset = 0x480F8) 495
8.12.1.70 at_dlradt_init_lut_index Register (offset = 0x480FC) 496
8.12.1.71 at_neg_delta Register (offset = 0x48148) 497
8.12.1.72 at_evt_enable Register (offset = 0x481F8) 498
8.12.1.73 at_evt_force Register (offset = 0x481FC) 499
8.12.1.74 at_internal_evt_enable Register (offset = 0x483F8) 500
8.12.1.75 at_internal_evt_force Register (offset = 0x483FC) 501
8.12.1.76 at_radt_sym_lut_ram[128] Register (offset = 0x4A000) 502
8.13 EE Registers 503
8.13.1 Grouped Common Registers Details 509
8.13.1.1 EE_VB_EOI Register (offset = 0x4000) 509
8.13.1.2 EE_VB_INTR_SET Register (offset = 0x4004) 510
8.13.1.3 EE_VB_INTR_CLR Register (offset = 0x4008) 511
8.13.1.4 ee_db_irs Register (offset = 0x4100) 512
8.13.1.5 ee_db_irs_set Register (offset = 0x4104) 513
8.13.1.6 ee_db_irs_clr Register (offset = 0x4108) 514
8.13.1.7 ee_db_en_ev0 Register (offset = 0x410C) 515
8.13.1.8 ee_db_en_set_ev0 Register (offset = 0x4110) 516
8.13.1.9 ee_db_en_clr_ev0 Register (offset = 0x4114) 517
8.13.1.10 ee_db_en_ev1 Register (offset = 0x4118) 518
8.13.1.11 ee_db_en_set_ev1 Register (offset = 0x411C) 519
8.13.1.12 ee_db_en_clr_ev1 Register (offset = 0x4120) 520
8.13.1.13 ee_db_en_sts_ev0 Register (offset = 0x4124) 521
8.13.1.14 ee_db_en_sts_ev1 Register (offset = 0x4128) 522
8.13.1.15 ee_ad_irs Register (offset = 0x4200) 523
8.13.1.16 ee_ad_irs_set Register (offset = 0x4204) 524
8.13.1.17 ee_ad_irs_clr Register (offset = 0x4208) 525
8.13.1.18 ee_ad_en_ev0 Register (offset = 0x420C) 526
8.13.1.19 ee_ad_en_set_ev0 Register (0x4210) 527
8.13.1.20 ee_ad_en_clr_ev0 Register (offset = 0x4214) 528
8.13.1.21 ee_ad_en_ev1 Register (offset = 0x4218) 529
8.13.1.22 ee_ad_en_set_ev1 Register (offset = 0x421C) 530
8.13.1.23 ee_ad_en_clr_ev1 Register (offset = 0x4220) 531

8.13.1.24 ee_ad_en_sts_ev0 Register (offset = 0x4224) 532
8.13.1.25 ee_ad_en_sts_ev1 Register (offset = 0x4228) 533
8.13.1.26 ee_cd_irs Register (offset = 0x4300) 534
8.13.1.27 ee_cd_irs_set Register (offset = 0x4304) 535
8.13.1.28 ee_cd_irs_clr Register (offset = 0x4308) 536
8.13.1.29 ee_cd_en_ev Register (offset = 0x430C) 537
8.13.1.30 ee_cd_en_set_ev Register (offset = 0x4310) 538
8.13.1.31 ee_cd_en_clr_ev Register (offset = x4314) 539
8.13.1.32 ee_cd_en_sts_ev Register (offset = 0x4318) 540
8.13.1.33 ee_sd_irs Register (offset = 0x4400) 541
8.13.1.34 ee_sd_irs_set Register (offset = 0x4404) 542
8.13.1.35 ee_sd_irs_clr Register (offset = 0x4408) 543
8.13.1.36 ee_sd_en_ev0 Register (offset = 0x440C) 544
8.13.1.37 ee_sd_en_set_ev0 Register (offset = 0x4410) 545
8.13.1.38 ee_sd_en_clr_ev0 Register (offset = 0x4414) 546
8.13.1.39 ee_sd_en_ev1 Register (offset = 0x4418) 547
8.13.1.40 ee_sd_en_set_ev1 Register (offset = 0x441C) 548
8.13.1.41 ee_sd_en_clr_ev1 Register (offset = 0x4420) 549
8.13.1.42 ee_sd_en_sts_ev0 Register (offset = 0x4424) 550
8.13.1.43 ee_sd_en_sts_ev1 Register (offset = 0x4428) 551
8.13.1.44 ee_vc_irs Register (offset = 0x4500) 552
8.13.1.45 ee_vc_irs_set Register (offset = 0x4504) 553
8.13.1.46 ee_vc_irs_clr Register (offset = 0x4508) 554
8.13.1.47 ee_vc_en_ev0 Register (offset = 0x450C) 555
8.13.1.48 ee_vc_en_set_ev0 Register (offset = 0x4510) 556
8.13.1.49 ee_vc_en_clr_ev0 Register (offset = 0x4514) 557
8.13.1.50 ee_vc_en_ev1 Register (offset = 0x4518) 558
8.13.1.51 ee_vc_en_set_ev1 Register (offset = 0x451C) 559
8.13.1.52 ee_vc_en_clr_ev1 Register (offset = 0x4520) 560
8.13.1.53 ee_vc_en_sts_ev0 Register (offset = 0x4524) 561
8.13.1.54 ee_vc_en_sts_ev1 Register (offset = 0x4528) 562
8.13.1.55 EE_AIF2_RUN_STS Register (offset = 0x4600) 563
8.13.1.56 EE_AIF2_RUN_CTL Register (offset = 0x4604) 564
8.13.1.57 ee_err_alrm_orgn Register (offset = 0x4700) 565
8.13.1.58 ee_lk_irs_a[0] Register (offset = 0x40000) 566
8.13.1.59 ee_lk_irs_set_a[0] Register (offset = 0x40004) 567
8.13.1.60 ee_lk_irs_clr_a[0] Register (offset = 0x40008) 568
8.13.1.61 ee_lk_en_a_ev0[0] Register (offset = 0x4000C) 569
8.13.1.62 ee_lk_en_a_set_ev0[0] Register (offset = 0x40010) 570
8.13.1.63 ee_lk_en_a_clr_ev0[0] Register (offset = 0x40014) 571
8.13.1.64 ee_lk_en_a_ev1[0] Register (offset = 0x40018) 572
8.13.1.65 ee_lk_en_a_set_ev1[0] Register (offset = 0x4001C) 573
8.13.1.66 ee_lk_en_a_clr_ev1[0] Register (offset = 0x40020) 574
8.13.1.67 ee_lk_en_sts_a_ev0[0] Register (offset = 0x40024) 575
8.13.1.68 ee_lk_en_sts_a_ev1[0] Register (offset = 0x40028) 576
8.13.1.69 ee_lk_irs_b[0] Register (offset = 0x4002C) 577
8.13.1.70 ee_lk_irs_set_b[0] Register (0x40030) 579
8.13.1.71 ee_lk_irs_clr_b[0] Register (offset = 0x40034) 580
8.13.1.72 ee_lk_en_b_ev0[0] Register (offset = 0x40038) 581
8.13.1.73 ee_lk_en_b_set_ev0[0] Register (offset = 0x4003C) 582
8.13.1.74 ee_lk_en_b_clr_ev0[0] Register (offset = 0x40040) 583
8.13.1.75 ee_lk_en_b_ev1[0] Register (offset = 0x40044) 584
8.13.1.76 ee_lk_en_b_set_ev1[0] Register (offset = 0x40048) 585

8.13.1.77 ee_lk_en_b_clr_ev1[0] Register (offset = 0x4004C) ... 586
8.13.1.78 ee_lk_en_sts_b_ev0[0] Register (offset = 0x40050) ... 587
8.13.1.79 ee_lk_en_sts_b_ev1[0] Register (offset = 0x40054) ... 588
8.13.1.80 ee_at_irs Register (offset = 0x40300) ... 589
8.13.1.81 ee_at_irs_set Register (offset = 0x40304) ... 590
8.13.1.82 ee_at_irs_clr Register (offset = 0x40308) ... 591
8.13.1.83 ee_at_en_ev0 Register (offset = 0x4030C) ... 592
8.13.1.84 ee_at_en_set_ev0 Register (offset = 0x40310) ... 593
8.13.1.85 ee_at_en_clr_ev0 Register (offset = 0x40314) ... 594
8.13.1.86 ee_at_en_ev1 Register (offset = 0x40318) ... 595
8.13.1.87 ee_at_en_set_ev1 Register (offset = 0x4031C) ... 596
8.13.1.88 ee_at_en_clr_ev1 Register (offset = 0x40320) ... 597
8.13.1.89 ee_at_en_sts_ev0 Register (offset = 0x40324) ... 598
8.13.1.90 ee_at_en_sts_ev1 Register (offset = 0x40328) ... 599
8.13.1.91 ee_pd_common_irs Register (offset = 0x40400) ... 600
8.13.1.92 ee_pd_common_irs_set Register (offset = 0x40404) ... 601
8.13.1.93 ee_pd_common_irs_clr Register (offset = 0x40408) ... 602
8.13.1.94 ee_pd_common_en_ev0 Register (offset = 0x4040C) ... 603
8.13.1.95 ee_pd_common_en_set_ev0 Register (offset = 0x40410) ... 604
8.13.1.96 ee_pd_common_en_clr_ev0 Register (offset = 0x40414) ... 605
8.13.1.97 ee_pd_common_en_ev1 Register (offset = 0x40418) ... 606
8.13.1.98 ee_pd_common_en_set_ev1 Register (offset = 0x4041C) ... 607
8.13.1.99 ee_pd_common_en_clr_ev1 Register (offset = 0x40420) ... 608
8.13.1.100 ee_pd_common_en_sts_ev0 Register (offset = 40424) ... 609
8.13.1.101 ee_pd_common_en_sts_ev1 Register (offset = 0x40428) ... 610
8.13.1.102 ee_pe_common_irs_set Register (offset = 0x40500) ... 611
8.13.1.103 ee_pe_common_irs_set Register (offset = 0x40504) ... 612
8.13.1.104 ee_pe_common_irs_clr Register (offset = 0x40508) ... 613
8.13.1.105 ee_pe_common_en_ev0 Register (offset = 0x4050C) ... 614
8.13.1.106 ee_pe_common_en_set_ev0 Register (offset = 0x40510) ... 615
8.13.1.107 ee_pe_common_en_clr_ev0 Register (offset = 0x40514) ... 616
8.13.1.108 ee_pe_common_en_ev1 Register (offset = 0x40518) ... 617
8.13.1.109 ee_pe_common_en_set_ev1 Register (offset = 0x4051C) ... 618
8.13.1.110 ee_pe_common_en_clr_ev1 Register (offset = 0x40520) ... 619
8.13.1.111 ee_pe_common_en_sts_ev0 Register (offset = 0x40254) ... 620
8.13.1.112 ee_pe_common_en_sts_ev1 Register (offset = 0x40528) ... 621

9 AIF2 Example Configuration ... 622

9.1 Transmission Rule Setup (PE Setup) ... 623
9.1.1 WCDMA OBSAI 4x, DL, 16 AxC With Four Control Channels (Four DSPs in Chain Topology) ... 623
9.1.2 WCDMA CPRI 4x, DL, 16 AxC With One Control Channel (Four DSPs in Chain Topology) ... 625
9.1.3 20 MHz LTE OBSAI 4x,DL, 2 AxC With Two Control Channels (Two DSPs in Chain Topology) ... 626
9.1.4 20 MHz LTE CPRI 4x, DL, 2 AxC With One Control Channel (Two DSPs in Chain Topology) ... 627
9.1.5 15 MHz LTE OBSAI With Two Different Options ... 628
9.2 Up Link Ingress Setup (PD, RT Setup) ... 629
9.2.1 WCDMA OBSAI 2x, 8 AxC With Retransmission ... 629
9.2.2 WCDMA CPRI 2x, 8 AxC With Aggregation ... 630
9.3 Direct IO Setup ... 631
9.3.1 WCDMA Two Ingress DIO Engine Setup For RAC A, B Broadcast For Three AxC Channels ... 631
9.3.2 WCDMA Egress DIO Engine Setup For TAC 32 AxC Channels ... 632
9.4 AIF2 PKTDMA and QM Setup ... 633
9.4.1 LTE 4x, 1 AxC (Channel 0) With Protocol-Specific Field ... 633
9.4.1.1 Queue Manager Setup ... 633
9.4.1.2 AIF2 PKTDMA Module Setup ... 635

9.4.1.3 Tx Queue Push Routine 637
9.5 Generic Packet Traffic Setup 638
9.5.1 OBSAI Generic Packet Traffic 638
9.5.2 CPRI Generic Packet Traffic 641
Revision History **644**

List of Figures

3-1. OFDM PKTDMA 42
3-2. LTE FDD Frame Structure 44
3-3. LTE TDD Frame Structure (5 ms Switch Point Illustrated) 44
3-4. CPRI LTE Sample Packing {20MHz, 10MHz, 5MHz} vs. {4x, 2x} Link Rate 47
3-5. TD-SCDMA Frame Structure 49
3-6. TD-SCDMA Example DL/UL Mix 49
3-7. TD-SCDMA AxCs Packed into Three CPRI Basic Frames 50
3-8. TD-SCDMA Slots Transferred To/From Memory 51
3-9. TD-SCDMA (AT) Timer-Based System Event Generation 52
3-10. WiMax Frame Structure 52
4-1. Timing Topology, Multiple RF, Common Alignment 61
4-2. Timing Topology, Multiple RF, Independent Alignment 62
4-3. OFDM/GSM/Edge2 Timing 62
4-4. WCDMA Timing 63
4-5. BTS AxC Offset 64
4-6. Pi and Delta Timing Example 65
4-7. OBSAI Radio Timing 66
4-8. Tx Framing Counter Circuit 66
4-9. How Control Messages Compress Time 69
4-10. Rx UL Timing, Daisy Chain Example 70
4-11. Valid Pi Window and Drift 71
4-12. AIF2 MAC Timing Measurements 76
4-13. WCDMA Egress Timing (OBSAI) 77
4-14. WCDMA Egress Timing (CPRI) 77
4-15. WCDMA Ingress Timing (OBSAI) 78
4-16. WCDMA Ingress Timing (CPRI) 78
4-17. LTE Egress Timing (OBSAI) 79
4-18. CLKSTP_REQ Min Timing 79
5-1. Ingress DMA Channel Addressing/Indexing 82
5-2. Ethernet Frame Structure 85
5-3. CPRI Packet Delimiters 88
5-4. OBSAI Data Format and Line Coding 92
5-5. OBSAI Frame/Message Structure 93
5-6. OBSAI Empty Message (AIF2 Transmitted) 94
5-7. OBSAI RP3 Transmit State Machine 94
5-8. OBSAI RP3 Receive State Machine 95
5-9. OBSAI Message, Payload Sample Packing 97
5-10. OBSAI DMA Index Circuit (Header-Based Reception) 98
5-11. OBSAI Transmission Rules 102
6-1. 16-bit Sample VBUS Transfer Format 105
6-2. Uplink VBUS Transfer Format 106
6-3. VBUS Format Generic (Byte Numbers Relative to PHY Arrival) 107
6-4. MMR VBUS_Config Data Format 108
6-5. Relation Between Frame Rate Event And 4chip Iteration Event 110
6-6. AIF2 Multicore Navigator 111
6-7. Multicore Navigator Host Mode Example 112
6-8. Multicore Navigator Monolithic Mode Example 112

7-1. AIF2 Hardware Architecture 122
7-2. AIF2 Clocking 125
7-3. OBSAI Serial Bit Ordering 126
7-4. CPRI Serial Bit Ordering 126
7-5. Infra-macro Case 130
7-6. Inter-macro Case 130
7-7. Clock Detect Watchdog 132
7-8. Clock Quality Measurement 132
7-9. Receive Pi Measurement (In AT Module) 133
7-10. Rx Mac Receive State Machine 135
7-11. Transmit State Machine 140
7-12. CPRI Data Format (2x link - 15-bit DL, 7-bit UL) 146
7-13. CPRI Data Format (5x link 7-bit UL) 147
7-14. CPRI Data Format (5x link 16-bit DL) 148
7-15. CPRI Data Format (4x link – 8-bit UL) 149
7-16. CPRI Internal Uplink Data Format (2x Link / 8-bit Data) 149
7-17. OBSAI DMA Index Circuit (Header-Based Reception) 151
7-18. PD_Pack_CPRI_Null_Delim Corner Case 155
7-19. PE/PD to DB internal Data Format 158
7-20. Window Wrap Cases 159
7-21. PD Frame : Framing Counter Circuit 160
7-22. PE: OBSAI Transmission Rules 163
7-23. PE: OBSAI Dual Bit Map Rules 164
7-24. PE: OBSAI Channel Look-Up, Part1 165
7-25. PE: OBSAI Channel Look-Up, Part 2 166
7-26. PE_TM_CPRI: Sample Rate Per Link Rate 168
7-27. Data Trace Quad-Word Byte Alignment (D0 First in Time) 180
7-28. Framing Data Quad-Word Alignment (S0 First in Time) 180
7-29. Buffer Channel Number Table (RAC Example: w/ 54 AxCs) 183
7-30. RAC Example 185
7-31. L2/RSA Example 186
7-32. DDR3 Eight Chip Block of Data 186
7-33. DDR3 Example 187
7-34. TAC Example 188
7-35. Timer Conceptual Diagram 191
7-36. AT Chip IO 192
7-37. PHYT Timer 193
7-38. RADT Basic Timer 194
7-39. RP1 Synchronization Burst Format 196
7-40. Payload Use for Type RP3 and WCDMA FDD 197
7-41. Payload Use for Type Time of Day 197
7-42. Event Generator 199
7-43. WCDMA RAD Timer Setup 201
7-44. WCDMA Event Per Slot (OBSAI) 201
7-45. WCDMA Event Every Four Chips (OBSAI) 201
7-46. LTE RAD Timer Setup 202
7-47. LTE TDD Event (OBSAI) 202
7-48. WIMAX Timer Setup 2 ms Frame, 22.4 MHz Sample Rate 203
7-49. WIMAX Events UL/DL Example 204

7-50. RADT TDSCDMA Counter Programming 204
7-51. TDSCDMA Events (CPRI) 205
7-52. GSM RADT Setup 206
7-53. GSM Mask Pattern 206
7-54. GSM T counters with Radio timer 207
7-55. EE Functional Block Diagram 208
7-56. Non-PKTDMA Error/Alarm Conditions 211
7-57. PKTDMA Error/Alarm Conditions 212

List of Tables

1-1. Abbreviations in the AIF2 User Guide 31
1-2. Definitions of Key Terms 33
3-1. LTE Channels Supported 43
3-2. LTE AxC Supported Over Different CPRI Link Rates 46
3-3. GSM Slot 307.2 MHz Approximations 54
3-4. OFDM On-The-Fly Update Requirements 57
4-1. QMSS Back-to-Back Performance 72
4-2. Egress (Tx) DIO Performance 73
4-3. Ingress (Rx) DIO Performance 73
4-4. AIF2 MAC Timing Measurements 76
5-1. Short Frame Length, OBSAI Message Group 84
5-2. Fast Ethernet 4B/5B Encoding 87
5-3. OBSAI RP3 SerDes Rates 92
5-4. OBSAI Data/Control Message Grouping 93
5-5. OBSAI RP3 Types 96
5-6. Scrambler Seed Values 101
6-1. Monolithic Packet Descriptor Word 0 115
6-2. Monolithic Packet Descriptor Word 1 116
6-3. Monolithic Packet Descriptor Word 2 116
6-4. Monolithic Packet Descriptor Word 3 (Protocol Specific Info) 117
6-5. Monolithic Packet Descriptor Word 3 (Protocol Specific Info Alternate Mode) 117
7-1. Emulation Suspend 123
7-2. Line Rate vs. PLL Output Clock Frequency 127
7-3. PLL Loop Bandwidth Selection (MHz) 128
7-4. Maximum allowable mon_wrap Value 132
7-5. RX Sync FSM State Transition 134
7-6. Rx Sync FSM Output 136
7-7. Rx Sync FSM State Names 136
7-8. Frame Sync State Machine Transition Conditions 140
7-9. Supported CPRI IQ Sample Widths 144
7-10. Supported CPRI Line Rates and Data Widths 145
7-11. RT Saturation Logic 150
7-12. OBSAI Dual Bit Map FSM Fields 165
7-13. PE_DB_IF AxC Watermark Settings 170
7-14. PE_DB_IF Packet Watermark Settings 170
7-15. DB Supported FIFO Buffer Sizes 173
7-16. DB_IDB(EDB)_PTR_CH Parameters for Programming Buffer Location and Size 174
7-17. DB_IDB_CFG_CHxx Parameters for Programming Alignment 174
7-18. Mapping of IDB Buffer Channel Enables 175
7-19. Debug Sideband Data Example for Packet Data 177
7-20. Debug Sideband Data Example for DirectIO Data 177
7-21. DB_IDB_CFG Parameters for Controlling Data Trace 180
7-22. Data Trace Token Types 181
7-23. Phy Timer Sync Selection 193
7-24. Rad Timer Sync Selection 194
7-25. RP1 Type Field Definition 196
7-26. Use of Timer Fields for Different Radio Standards 200

7-27. MMR General Description 210
7-28. ERR_ALARM_ORGN Bit Definitions 213
8-1. VC Register Memory Map 217
8-2. AIF2 Peripheral ID Register Field Descriptions 217
8-3. AIF2 Scratch Register Field Descriptions 218
8-4. AIF2 Software Reset Register Field Descriptions 219
8-5. AIF2 Emulation Control Register Field Descriptions 220
8-6. VC Status Register Field Descriptions 221
8-7. SD Register Memory Map 222
8-8. SD_RX_EN_CFG[0] Register Field Descriptions 224
8-9. SD_RX_R1_CFG[0] Register Field Descriptions 225
8-10. SD_RX_R2_CFG[0] Register Field Descriptions 226
8-11. SD_RX_STS[0] Register Field Descriptions 227
8-12. SD_TX_EN_CFG[0] Register Field Descriptions 228
8-13. SD_TX_R1_CFG[0] Register Field Descriptions 229
8-14. SD_TX_R2_CFG[0] Register Field Descriptions 230
8-15. SD_TX_STS[0] Register Field Descriptions 232
8-16. SD_PLL_B8_EN_CFG Register Field Descriptions 233
8-17. SD_PLL_B4_EN_CFG Register Field Descriptions 234
8-18. SD_PLL_B8_CFG Register Field Descriptions 235
8-19. SD_PLL_B4_CFG Register Field Descriptions 236
8-20. SD_PLL_B8_STS Register Field Descriptions 237
8-21. SD_PLL_B4_STS Register Field Descriptions 238
8-22. SD_CLK_SEL_CFG Register Field Descriptions 239
8-23. SD_LK_CLK_DIS_CFG Register Field Descriptions 240
8-24. RM Register Memory Map 241
8-25. RM_LK_CFG0[0] Register Field Descriptions 243
8-26. RM_LK_CFG1[0] Register Field Descriptions 244
8-27. RM_LK_CFG2[0] Register Field Descriptions 245
8-28. RM_LK_CFG3[0] Register Field Descriptions 246
8-29. RM_LK_CFG4[0] Register Field Descriptions 247
8-30. RM_LK_STS0[0] Register Field Descriptions 248
8-31. RM_LK_STS1[0] Register Field Descriptions 249
8-32. RM_LK_STS2[0] Register Field Descriptions 250
8-33. RM_LK_STS3[0] Register Field Descriptions 251
8-34. RM_LK_STS4[0] Register Field Descriptions 252
8-35. RM_CFG Register Field Descriptions 253
8-36. TM Register Memory Map 254
8-37. TM_LK_CFG[0] Register Field Descriptions 258
8-38. TM_LK_CTRL[0] Register Field Descriptions 259
8-39. TM_LK_SCR_CTRL[0] Register Field Descriptions 260
8-40. TM_LK_L1_CFG[0] Register Field Descriptions 261
8-41. TM_LK_L1_EN[0] Register Field Descriptions 262
8-42. TM_LK_LOSERR[0] Register Field Descriptions 263
8-43. TM_LK_LOFERR[0] Register Field Descriptions 264
8-44. TM_LK_LOSRx[0] Register Field Descriptions 265
8-45. TM_LK_LOFRx[0] Register Field Descriptions 266
8-46. TM_LK_RAIRx[0] Register Field Descriptions 267
8-47. TM_LK_HFN[0] Register Field Descriptions 268

8-48. TM_LK_PTRP[0] Register Field Descriptions 269
8-49. TM_LK_STRT[0] Register Field Descriptions 270
8-50. TM_LK_PROT[0] Register Field Descriptions 271
8-51. TM_LK_STAT[0] Register Field Descriptions 272
8-52. TM_FRM_MODE[0] Register Field Descriptions 273
8-53. CI Register Memory Map 274
8-54. CI_LK_CFG[0] Register Field Descriptions 274
8-55. CO Register Memory Map 275
8-56. CO_LK_CFG[0] Register Field Descriptions 275
8-57. RT Register Memory Map 276
8-58. RT_LK_CFG[0] Register Field Descriptions 277
8-59. RT_LK_DPTH[0] Register Field Descriptions 278
8-60. RT_HDR_ERR[0] Register Field Descriptions 279
8-61. RT_LK_STAT[0] Register Field Descriptions 280
8-62. PD Register Memory Map 281
8-63. pd_link_a[0] Register Field Descriptions 285
8-64. pd_link_b[0] Register Field Descriptions 286
8-65. pd_lk_pack_cpri[0] Register Field Descriptions 287
8-66. pd_cpri_crc[0] Register Field Descriptions 288
8-67. pd_pack_map[0] Register Field Descriptions 289
8-68. pd_dbm[0] Register Field Descriptions 290
8-69. pd_dbm_1map[0][0-3] Register Field Descriptions 291
8-70. pd_dbm_2map[0][0-2] Register Field Descriptions 292
8-71. pd_type_lut[0][0-31] Register Field Descriptions 293
8-72. pd_cpri_id_lut[0][0-127] Register Field Descriptions 294
8-73. pd_cw_lut[0][0-255] Register Field Descriptions 295
8-74. pd_global Register Field Descriptions 296
8-75. pd_global_en_set Register Field Descriptions 297
8-76. pd_global_en_clr Register Field Descriptions 298
8-77. pd_global_en_sts Register Field Descriptions 299
8-78. pd_dma Register Field Descriptions 300
8-79. pd_radt_tc Register Field Descriptions 301
8-80. pd_chan_sts[4] Register Field Descriptions 302
8-81. pd_pkt_sts[4] Register Field Descriptions 303
8-82. pd_frm_tc[6] Register Field Descriptions 304
8-83. pd_route[128] Register Field Descriptions 305
8-84. pd_dmachan[128] Register Field Descriptions 306
8-85. pd_dmachan_a[128] Register Field Descriptions 307
8-86. pd_dmachan_b[128] Register Field Descriptions 308
8-87. pd_dmachan_c[128] Register Field Descriptions 309
8-88. pd_dmachan_d[128] Register Field Descriptions 310
8-89. pd_dmachan_e[128] Register Field Descriptions 311
8-90. pd_dmachan_f[128] Register Field Descriptions 312
8-91. pd_frm_msg_tc[128] Register Field Descriptions 313
8-92. PE Register Memory Map 314
8-93. pe_link[0] Register Field Descriptions 317
8-94. pe_crc[0] Register Field Descriptions 318
8-95. pe_cpri_dbm[0] Register Field Descriptions 319
8-96. pe_cpri_bitswap[0] Register Field Descriptions 320

8-97. pe_cpridbm_1map[0][0-3] Register Field Descriptions 321
8-98. pe_cpridbm_2map[0][0-2] Register Field Descriptions 322
8-99. pe_cpri0[0] Register Field Descriptions 323
8-100. pe_cpri1[0] Register Field Descriptions 324
8-101. pe_cw_lut[0][0-255] Register Field Descriptions 325
8-102. pe_global Register Field Descriptions 326
8-103. pe_global_en_set Register Field Descriptions 327
8-104. pe_global_en_clr Register Field Descriptions 327
8-105. pe_global_en_sts Register Field Descriptions 328
8-106. pe_chan_sts[4] Register Field Descriptions 329
8-107. pe_pkt_sts[4] Register Field Descriptions 330
8-108. pe_frm_tc[6] Register Field Descriptions 331
8-109. pe_dmachan_en[128] Register Field Descriptions 332
8-110. pe_dma0chan[128] Register Field Descriptions 333
8-111. pe_in_fifo[128] Register Field Descriptions 334
8-112. pe_axc_offset[128] Register Field Descriptions 335
8-113. pe_frm_msg_tc[128] Register Field Descriptions 336
8-114. pe_modtxrule[64] Register Field Descriptions 337
8-115. pe_obsai_hdr[0-127] Register Field Descriptions 338
8-116. pe_obsai_dbm[64] Register Field Descriptions 339
8-117. pe_dbm_map[512] Register Field Descriptions 340
8-118. pe_rule_chanlut0[512] Register Field Descriptions 341
8-119. pe_rule_chanlut1[512] Register Field Descriptions 342
8-120. pe_rule_chanlut2[512] Register Field Descriptions 343
8-121. pe_rule_chanlut3[512] Register Field Descriptions 344
8-122. pe_rule_chanlut4[512] Register Field Descriptions 345
8-123. pe_rule_chanlut5[512] Register Field Descriptions 346
8-124. pe_rule_chanlut6[512] Register Field Descriptions 347
8-125. pe_rule_chanlut7[512] Register Field Descriptions 348
8-126. DB Register Memory Map 349
8-127. DB_IDB_CFG Register Field Descriptions 351
8-128. DB_IDB_GLOBAL_EN_SET Register Field Descriptions 352
8-129. DB_IDB_GLOBAL_EN_CLR Register Field Descriptions 353
8-130. DB_IDB_GLOBAL_EN_STS Register Field Descriptions 354
8-131. DB_IDB_CH_EN[4] Register Field Descriptions 355
8-132. DB_IDB_DEBUG_D0 Register Field Descriptions 356
8-133. DB_IDB_DEBUG_D1 Register Field Descriptions 357
8-134. DB_IDB_DEBUG_D2 Register Field Descriptions 358
8-135. DB_IDB_DEBUG_D3 Register Field Descriptions 359
8-136. DB_IDB_DEBUG_SBND Register Field Descriptions 360
8-137. DB_IDB_DEBUG_DB_WR Register Field Descriptions 361
8-138. DB_IDB_DEBUG_OFS Register Field Descriptions 362
8-139. DB_IDB_DEBUG_OFS_DAT Register Field Descriptions 363
8-140. DB_IDB_PTR_CH[128] Register Field Descriptions 364
8-141. DB_IDB_CFG_CH[128] Register Field Descriptions 365
8-142. DB_IDB_CH_EMPTY[4] Register Field Descriptions 366
8-143. DB_EDB_CFG Register Field Descriptions 367
8-144. DB_EDB_GLOBAL_EN_SET Register Field Descriptions 368
8-145. DB_EDB_GLOBAL_EN_CLR Register Field Descriptions 369

8-146. DB_EDB_GLOBAL_EN_STS Register Field Descriptions 370
8-147. DB_EDB_CH_EN[4] Register Field Descriptions 371
8-148. DB_EDB_DEBUG_D0 Register Field Descriptions 372
8-149. DB_EDB_DEBUG_D1 Register Field Descriptions 373
8-150. DB_EDB_DEBUG_D2 Register Field Descriptions 374
8-151. DB_EDB_DEBUG_D3 Register Field Descriptions 375
8-152. DB_EDB_DEBUG_SBND Register Field Descriptions 376
8-153. DB_EDB_DEBUG_RD_CNTL Register Field Descriptions 377
8-154. DB_EDB_DEBUG_DB_RD Register Field Descriptions 378
8-155. DB_EDB_DEBUG_OFS Register Field Descriptions 379
8-156. DB_EDB_DEBUG_OFS_DAT Register Field Descriptions 380
8-157. DB_EDB_DEBUG_WR_TOK Register Field Descriptions 381
8-158. DB_EDB_EOP_CNT Register Field Descriptions 382
8-159. DB_EDB_PTR_CH[128] Register Field Descriptions 383
8-160. DB_EDB_CFG_CH[128] Register Field Descriptions 384
8-161. AD Register Memory Map 385
8-162. AD_DIO_I_TABLE_SEL[0] Register Field Descriptions 388
8-163. AD_DIO_I_TABLE_LOOP_CFG[0] Register Field Descriptions 389
8-164. AD_DIO_I_DMA_CFG0[0] Register Field Descriptions 390
8-165. AD_DIO_I_DMA_CFG1[0] Register Field Descriptions 391
8-166. AD_DIO_I_DMA_CFG2[0] Register Field Descriptions 392
8-167. AD_DIO_I_BCN_TABLE0_ROW0[0] to AD_DIO_I_BCN_TABLE0_ROW15[0] Register Field Descriptions 393
8-168. AD_DIO_I_BCN_TABLE1_ROW0[0] to AD_DIO_I_BCN_TABLE1_ROW15[0] Register Field Descriptions 394
8-169. AD_DIO_E_TABLE_SEL[0] Register Field Descriptions 395
8-170. AD_DIO_E_TABLE_LOOP_CFG[0] Register Field Descriptions 396
8-171. AD_DIO_E_DMA_CFG0[0] Register Field Descriptions 397
8-172. AD_DIO_E_DMA_CFG1[0] Register Field Descriptions 398
8-173. AD_DIO_E_DMA_CFG2[0] Register Field Descriptions 399
8-174. AD_DIO_E_BCN_TABLE0_ROW0[0] to AD_DIO_E_BCN_TABLE0_ROW15[0] Register Field Descriptions 400
8-175. AD_DIO_E_BCN_TABLE1_ROW0[0] to AD_DIO_E_BCN_TABLE1_ROW15[0] Register Field Descriptions 401
8-176. AD_DIO_DT_DMA_CFG0 Register Field Descriptions 402
8-177. AD_DIO_DT_DMA_CFG1 Register Field Descriptions 403
8-178. AD_DIO_DT_DMA_CFG2 Register Field Descriptions 404
8-179. AD_DIO_DT_DMA_CFG3n Register Field Descriptions 405
8-180. AD_DIO_I_GLOBAL_EN_SET Register Field Descriptions 406
8-181. AD_ DIO_I _GLOBAL_EN_CLR Register Field Descriptions 407
8-182. AD_ DIO_I _GLOBAL_EN_STS Register Field Descriptions 408
8-183. AD_DIO_E_GLOBAL_EN_SET Register Field Descriptions 409
8-184. AD_ DIO_E _GLOBAL_EN_CLR Register Field Descriptions 410
8-185. AD_ DIO_E_GLOBAL_EN_STS Register Field Descriptions 411
8-186. AD_ISCH_CFG Register Field Descriptions 412
8-187. AD_ISCH_GLOBAL_EN_SET Register Field Descriptions 413
8-188. AD_ISCH_GLOBAL_EN_CLR Register Field Descriptions 414
8-189. AD_ISCH_GLOBAL_EN_STS Register Field Descriptions 415
8-190. AD_ISCH_EOP_CNT Register Field Descriptions 416
8-191. AD_ESCH_CFG Register Field Descriptions 417
8-192. AD_ESCH_GLOBAL_EN_SET Register Field Descriptions 418

8-193. AD_ESCH_GLOBAL_EN_CLR Register Field Descriptions 419
8-194. AD_ESCH_GLOBAL_EN_STS Register Field Descriptions 420
8-195. AT Register Memory Map 421
8-196. at_pimax_lk[0] Register Field Descriptions 427
8-197. at_pimin_lk[0] Register Field Descriptions 428
8-198. at_pivalue_lk[0] Register Field Descriptions 429
8-199. at_event_offset[0] Register Field Descriptions 430
8-200. at_event_mod_tc[0] Register Field Descriptions 431
8-201. at_event_mask_lsbs[0] Register Field Descriptions 432
8-202. at_event_mask_msbs[0] Register Field Descriptions 433
8-203. at_ad_ingr_event_offset[0] Register Field Descriptions 434
8-204. at_ad_ingr_event_mod_tc[0] Register Field Descriptions 435
8-205. at_ad_egr_event_offset[0] Register Field Descriptions 436
8-206. at_ad_egr_event_mod_tc[0] Register Field Descriptions 437
8-207. at_tm_delta_event_offset[0] Register Field Descriptions 438
8-208. at_tm_delta_event_mod_tc[0] Register Field Descriptions 439
8-209. at_pe_event_offset[0] Register Field Descriptions 440
8-210. at_pe_event_mod_tc[0] Register Field Descriptions 441
8-211. at_pe_event2_offset[0] Register Field Descriptions 442
8-212. at_pe_event2_mod_tc[0] Register Field Descriptions 443
8-213. at_control1 Register Field Descriptions 444
8-214. at_control2 Register Field Descriptions 445
8-215. at_sw_sync Register Field Descriptions 446
8-216. at_phyt_cmp_radsync Register Field Descriptions 447
8-217. at_rp1_type Register Field Descriptions 448
8-218. at_captradt Register Field Descriptions 449
8-219. at_rp1_type_capture Register Field Descriptions 450
8-220. at_rp1_tod_capture_l Register Field Descriptions 451
8-221. at_rp1_tod_capture_h Register Field Descriptions 452
8-222. at_rp1_rp3_capture_l Register Field Descriptions 453
8-223. at_rp1_rp3_capture_h Register Field Descriptions 454
8-224. at_rp1_rad_capture_l Register Field Descriptions 455
8-225. at_rp1_rad_capture_h Register Field Descriptions 456
8-226. at_phyt_clkcnt_value Register Field Descriptions 457
8-227. at_phyt_frm_value_lsbs Register Field Descriptions 458
8-228. at_phyt_frm_value_msbs Register Field Descriptions 459
8-229. at_radt_value_lsbs Register Field Descriptions 460
8-230. at_radt_value_mid Register Field Descriptions 461
8-231. at_radt_value_msbs Register Field Descriptions 462
8-232. at_ulradt_value_lsbs Register Field Descriptions 463
8-233. at_ulradt_value_mid Register Field Descriptions 464
8-234. at_ulradt_value_msbs Register Field Descriptions 465
8-235. at_dlradt_value_lsbs Register Field Descriptions 466
8-236. at_dlradt_value_mid Register Field Descriptions 467
8-237. at_dlradt_value_msbs Register Field Descriptions 468
8-238. at_radt_wcdma_value Register Field Descriptions 469
8-239. at_ulradt_wcdma_value Register Field Descriptions 470
8-240. at_dlradt_wcdma_value Register Field Descriptions 471
8-241. at_radt_wcdma_div Register Field Descriptions 472

8-242. at_phyt_init_lsbs Register Field Descriptions 473
8-243. at_phyt_init_mid Register Field Descriptions 474
8-244. at_phyt_init_msbs Register Field Descriptions 475
8-245. at_phyt_tc_lsbs Register Field Descriptions 476
8-246. at_phyt_frame_tc_lsbs Register Field Descriptions 477
8-247. at_phyt_frame_tc_msbs Register Field Descriptions 478
8-248. at_radt_init_lsbs Register Field Descriptions 479
8-249. at_radt_init_mid Register Field Descriptions 480
8-250. at_radt_init_msbs Register Field Descriptions 481
8-251. at_ulradt_init_lsbs Register Field Descriptions 482
8-252. at_radt_tstamp_value Register Field Descriptions 483
8-253. at_dlradt_init_lsbs Register Field Descriptions 484
8-254. at_gsm_tcount_init Register Field Descriptions 485
8-255. at_gsm_tcount_value Register Field Descriptions 486
8-256. at_radt_symb_lut_index_tc Register 487
8-257. at_radt_frame_tc_lsbs Register Field Descriptions 488
8-258. at_radt_frame_tc_msbs Register Field Descriptions 489
8-259. at_ulradt_init_mid Register Field Descriptions 490
8-260. at_ulradt_init_msbs Register Field Descriptions 491
8-261. at_dlradt_init_mid Register Field Descriptions 492
8-262. at_dlradt_init_msbs Register Field Descriptions 493
8-263. at_radt_init_lut_index Register Field Descriptions 494
8-264. at_ulradt_init_lut_index Register Field Descriptions 495
8-265. at_dlradt_init_lut_index Register Field Descriptions 496
8-266. at_neg_delta Register Field Descriptions 497
8-267. at_evt_enable Register Field Descriptions 498
8-268. at_evt_force Register Field Descriptions 499
8-269. at_internal_evt_enable Register Field Descriptions 500
8-270. at_internal_evt_force Register Field Descriptions 501
8-271. at_radt_sym_lut_ram[128] Register Field Descriptions 502
8-272. EE Register Memory Map 503
8-273. EE_VB_EOI Register Field Descriptions 509
8-274. EE_VB_INTR_SET Register Field Descriptions 510
8-275. EE_VB_INTR_CLR Register Field Descriptions 511
8-276. ee_db_irs Register Field Descriptions 512
8-277. ee_db_irs_set Register Field Descriptions 513
8-278. ee_db_irs_clr Register Field Descriptions 514
8-279. ee_db_en_ev0 Register Field Descriptions 515
8-280. ee_db_en_set_ev0 Register Field Descriptions 516
8-281. ee_db_en_clr_ev0 Register Field Descriptions 517
8-282. ee_db_en_ev1 Register Field Descriptions 518
8-283. ee_db_en_set_ev1 Register Field Descriptions 519
8-284. ee_db_en_clr_ev1 Register Field Descriptions 520
8-285. ee_db_en_sts_ev0 Register Field Descriptions 521
8-286. ee_db_en_sts_ev1 Register Field Descriptions 522
8-287. ee_ad_irs Register Field Descriptions 523
8-288. ee_ad_irs_set Register Field Descriptions 524
8-289. ee_ad_irs_clr Register Field Descriptions 525
8-290. ee_ad_en_ev0 Register Field Descriptions 526

8-291. ee_ad_en_set_ev0 Register Field Descriptions 527
8-292. ee_ad_en_clr_ev0 Register Field Descriptions 528
8-293. ee_ad_en_ev1 Register Field Descriptions 529
8-294. ee_ad_en_set_ev1 Register Field Descriptions 530
8-295. ee_ad_en_clr_ev1 Register Field Descriptions 531
8-296. ee_ad_en_sts_ev0 Register Field Descriptions 532
8-297. ee_ad_en_sts_ev1 Register Field Descriptions 533
8-298. ee_cd_irs Register Field Descriptions 534
8-299. ee_cd_irs_set Register Field Descriptions 535
8-300. ee_cd_irs_clr Register Field Descriptions 536
8-301. ee_cd_en_ev Register Field Descriptions 537
8-302. ee_cd_en_set_ev Register Field Descriptions 538
8-303. ee_cd_en_clr_ev Register Field Descriptions 539
8-304. ee_cd_en_sts_ev Register Field Descriptions 540
8-305. ee_sd_irs Register Field Descriptions 541
8-306. ee_sd_irs_set Register Field Descriptions 542
8-307. ee_sd_irs_clr Register Field Descriptions 543
8-308. ee_sd_en_ev0 Register Field Descriptions 544
8-309. ee_sd_en_set_ev0 Register Field Descriptions 545
8-310. ee_sd_en_clr_ev0 Register Field Descriptions 546
8-311. ee_sd_en_ev1 Register Field Descriptions 547
8-312. ee_sd_en_set_ev1 Register Field Descriptions 548
8-313. ee_sd_en_clr_ev1 Register Field Descriptions 549
8-314. ee_sd_en_sts_ev0 Register Field Descriptions 550
8-315. ee_sd_en_sts_ev1 Register Field Descriptions 551
8-316. ee_vc_irs Register Field Descriptions 552
8-317. ee_vc_irs_set Register Field Descriptions 553
8-318. ee_vc_irs_clr Register Field Descriptions 554
8-319. ee_vc_en_ev0 Register Field Descriptions 555
8-320. ee_vc_en_set_ev0 Register Field Descriptions 556
8-321. ee_vc_en_clr_ev0 Register Field Descriptions 557
8-322. ee_vc_en_ev1 Register Field Descriptions 558
8-323. ee_vc_en_set_ev1 Register Field Descriptions 559
8-324. ee_vc_en_clr_ev1 Register Field Descriptions 560
8-325. ee_vc_en_sts_ev0 Register Field Descriptions 561
8-326. ee_vc_en_sts_ev1 Register Field Descriptions 562
8-327. EE_AIF2_RUN_STS Register Field Descriptions 563
8-328. EE_AIF2_RUN_CTL Register Field Descriptions 564
8-329. ee_err_alrm_orgn Register Field Descriptions 565
8-330. ee_lk_irs_a[0] Register Field Descriptions 566
8-331. ee_lk_irs_set_a[0] Register Field Descriptions 567
8-332. ee_lk_irs_clr_a[0] Register Field Descriptions 568
8-333. ee_lk_en_a_ev0[0] Register Field Descriptions 569
8-334. ee_lk_en_a_set_ev0[0] Register Field Descriptions 570
8-335. ee_lk_en_a_clr_ev0[0] Register Field Descriptions 571
8-336. ee_lk_en_a_ev1[0] Register Field Descriptions 572
8-337. ee_lk_en_a_set_ev1[0] Register Field Descriptions 573
8-338. ee_lk_en_a_clr_ev1[0] Register Field Descriptions 574
8-339. ee_lk_en_sts_a_ev0[0] Register Field Descriptions 575

8-340. ee_lk_en_sts_a_ev1[0] Register Field Descriptions 576
8-341. ee_lk_irs_b[0] Register Field Descriptions 577
8-342. ee_lk_irs_set_b[0] Register Field Descriptions 579
8-343. ee_lk_irs_clr_b[0] Register Field Descriptions 580
8-344. ee_lk_en_b_ev0[0] Register Field Descriptions 581
8-345. ee_lk_en_b_set_ev0[0] Register Field Descriptions 582
8-346. ee_lk_en_b_clr_ev0[0] Register Field Descriptions 583
8-347. ee_lk_en_b_ev1[0] Register Field Descriptions 584
8-348. ee_lk_en_b_set_ev1[0] Register Field Descriptions 585
8-349. ee_lk_en_b_clr_ev1[0] Register Field Descriptions 586
8-350. ee_lk_en_sts_b_ev0[0] Register Field Descriptions 587
8-351. ee_lk_en_sts_b_ev1[0] Register Field Descriptions 588
8-352. ee_at_irs Register Field Descriptions 589
8-353. ee_at_irs_set Register Field Descriptions 590
8-354. ee_at_irs_clr Register Field Descriptions 591
8-355. ee_at_en_ev0 Register Field Descriptions 592
8-356. ee_at_en_set_ev0 Register Field Descriptions 593
8-357. ee_at_en_clr_ev0 Register Field Descriptions 594
8-358. ee_at_en_ev1 Register Field Descriptions 595
8-359. ee_at_en_set_ev1 Register Field Descriptions 596
8-360. ee_at_en_clr_ev1 Register Field Descriptions 597
8-361. ee_at_en_sts_ev0 Register Field Descriptions 598
8-362. ee_at_en_sts_ev1 Register Field Descriptions 599
8-363. ee_pd_common_irs Register Field Descriptions 600
8-364. ee_pd_common_irs_set Register Field Descriptions 601
8-365. ee_pd_common_irs_clr Register Field Descriptions 602
8-366. ee_pd_common_en_ev0 Register Field Descriptions 603
8-367. ee_pd_common_en_set_ev0 Register Field Descriptions 604
8-368. ee_pd_common_en_clr_ev0 Register Field Descriptions 605
8-369. ee_pd_common_en_ev1 Register Field Descriptions 606
8-370. ee_pd_common_en_set_ev1 Register Field Descriptions 607
8-371. ee_pd_common_en_clr_ev1 Register Field Descriptions 608
8-372. ee_pd_common_en_sts_ev0 Register Field Descriptions 609
8-373. ee_pd_common_en_sts_ev1 Register Field Descriptions 610
8-374. ee_pe_common_irs_set Register Field Descriptions 611
8-375. ee_pe_common_irs_set Register Field Descriptions 612
8-376. ee_pe_common_irs_clr Register Field Descriptions 613
8-377. ee_pe_common_en_ev0 Register Field Descriptions 614
8-378. ee_pe_common_en_set_ev0 Register Field Descriptions 615
8-379. ee_pe_common_en_clr_ev0 Register Field Descriptions 616
8-380. ee_pe_common_en_ev1 Register Field Descriptions 617
8-381. ee_pe_common_en_set_ev1 Register Field Descriptions 618
8-382. ee_pe_common_en_clr_ev1 Register 619
8-383. ee_pe_common_en_sts_ev0 Register Field Descriptions 620
8-384. ee_pe_common_en_sts_ev1 Register Field Descriptions 621

Preface

About This Manual

Antenna Interface 2 (AIF2) is a peripheral module that supports transfers of baseband IQ data between uplink and downlink baseband DSP processors and a high-speed serial interface.

Notational Conventions

This document uses the following conventions:

- Commands and keywords are in **boldface** font.
- Arguments for which you supply values are in *italic* font.
- Terminal sessions and information the system displays are in screen font.
- Information you must enter is in **boldface screen font**.
- Elements in square brackets ([ ]) are optional.

Notes use the following conventions:

NOTE: Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

CAUTION

Indicates the possibility of service interruption if precautions are not taken.

WARNING

Indicates the possibility of damage to equipment if precautions are not taken.

Related Documentation from Texas Instruments

AIF1-to-AIF2 Antenna Interface Migration Guide for KeyStone Devices	SPRABH8
Connecting AIF2 with FFTC	SPRABF3
DDR3 Memory Controller for KeyStone Devices User Guide	SPRUGV8
Multicore Navigator for KeyStone Devices User Guide	SPRUGR9
Multicore Shared Memory Controller (MSMC) for KeyStone Devices User Guide	SPRUGW7

Introduction

NOTE: The information in this document should be used in conjunction with information in the device-specific Keystone Architecture data manual that applies to the part number of your device.

Topic **Page**

1.1 **Purpose** 31
1.2 **Scope** 31
1.3 **Terminology** 31
1.4 **Features** 33

1.1 Purpose

Antenna Interface 2 (AIF2) is a peripheral module that supports transfers of baseband IQ data between uplink and downlink baseband DSP processors and a high-speed serial interface.

This document describes

- How to use AIF2
- Supported features
- Interface and implementation details
- AIF2 memory mapped registers

1.2 Scope

The scope of this document is limited to the AIF2 IP. This version of the document has a large amount of information about internal AIF2 modules and features. It also includes information about Multicore Navigator and shows some examples of how to use it but does not include detailed configuration of AIF2 PKTDMA and QM.

1.3 Terminology

The following sections list abbreviations, acronyms, and definitions of key terms that appear in this user guide.

1.3.1 Abbreviations and Acronyms

Table 1-1. Abbreviations in the AIF2 User Guide

Abbreviation	Description
AD	AIF2 DMA
AIF2	Antenna Interface 2
AT	AIF2 timer
AxC	Antenna carrier
AxS	Antenna carrier sample
BER	SerDes bit error rate
BFN	UMTS nodeB frame number
C&M	Control and management
CDMA	CPPI DMA module (now called PKTDMA)
CFG	Configuration
CI	Common public radio interface input data
CO	Common public radio interface output data
CorePac	A specific DSP core used in the KeyStone architecture
CPPI	Common Port Programming Interface (now called Multicore Navigator)
CPRI	Common Public Radio Interface
CRC	Cyclic redundancy check
CSL	Chip support library
CW	Control Word (CPRI)
DB	Data buffer
DDR3	Dual data rate memory 3
DL	Down link
DMA	Direct memory access
DSP	Digital signal processor
EE	Exception event handler
FDD	Frequency division duplex
FIFO	First in first out queue memory structure

Table 1-1. Abbreviations in the AIF2 User Guide (continued)

Abbreviation	Description
FSM	Finite state machine
HFN	Hyper frame number
I/F	Interface
IO	Input and output data flow
IQ	In-phase and quadrature data
L2	CorePac DSP level 2 SRAM
LOF	Loss of frame
LOS	Loss of signal
LSB	Least significant bit
LTE	Long term evolution
LUT	Look up table
MMR	Memory mapped register
MSB	Most significant bit
Multicore Navigator	(formerly CPPI)
OBSAI	Open Base Station Architecture Initiative
PD	Protocol decoder
PE	Protocol encoder
PKTDMA	Packet DMA (formerly CPPI DMA and CDMA)
PLL	Phase lock loop
PS	Packet switched
QM	Queue Manager
RAC	Receive accelerator co-processor
RAI	Remote alarm indication
RAM	Random access memory
RF	Radio frequency
RM	Receive Media access control
RP1	Reference point 1 (OBSAI)
RP3	Reference point 3 (OBSAI)
RT	Retransmitter
RX	Receive
SD	Serializer / Deserializer (SerDes)
TAC	Transmit accelerator co-processor
TM	Transmit Media access control
TX	Transmit
UL	Uplink
UMTS	Universal mobile telecommunication system
VBUSM	Virtual bus multi-issue
VBUSP	Virtual bus pipeline
VC	VBUSP configuration bus interface
WCDMA	Wideband code division multiple access

1.3.2 Definitions

Table 1-2. Definitions of Key Terms

Term	Definition
Channel	(same as stream)
Chip	WCDMA term indicating a sample of time. In WCDMA, one chip represents one clock cycle of 3.84MHz (appox 260 ns). Term is also used to indicate a 16-bit I /16-bit Q sample, or two 8-bit I/ 8-bit Q samples for UL 2x oversampled.
Egress	Data or control information that has or will **exit** DSP
Ingress	Data or control information that has or will **enter** DSP
Link	A link is equivalent to a SerDes channel and the associated OBSAI or CPRI protocol that passes over that link. Links have an associated bitrate where the bitrate has some degree of programmability.
PKTDMA Packet	A PKTDMA packet consists of a PKTDMA header (called a descriptor) plus payload information (called a buffer) and optional protocol-specific data, which is an extension of payload information.
Stream	AIF2, CPRI, and OBSAI have the concept of multiple parallel contexts interleaved or arbitrated onto a single link. Stream is a term that is sometimes used to identify a single context (or series of packets). Channel is also used to identify the same concept. A given stream can either be streaming such as with ADC antenna data or on-demand as in packet date. AIF2 supports ingress and egress streams. 128 streams are supported for each. The concept of stream is very important as AIF2 has unpacking and packing buffers dedicated per stream.
Wrap	AIF2 has circular ram, terminal count, and LUT; these fields use their own limited count value. If the count value reaches the terminal count value, it will automatically go back to the initial value or zero value. Wrap means this kind of circular repeating activity of a register field or memory.

1.4 Features

The following features are supported:

- Supported Standards
  - OBSAI
    - OBSAI WCDMA
    - OBSAI LTE
    - OBSAI WiMax (all 802.16 formats)
    - OBSAI GSM/Edge
    - OBSAI Generic Packet Traffic
  - CPRI
    - CPRI WCDMA
    - CPRI LTE
    - CPRI WiMax (all 802.16 formats)
    - CPRI TDSCDMA support
    - CPRI Control Word (Generic Traffic)
- Six Links
  - 6 GHz SerDes
  - 6 GHz SerDes bit-level scrambling (OBSAI specified)
  - Independent Link rate per link—(exception is CPRI 5x rate, all or none)
  - Virtual Link support: (that is, 4x rate link can support four 1x virtual links)
  - Link Rates
    - OBSAI {2x, 4x, 8x}
    - CPRI {2x, 4x, 5x, 8x}

- Antenna Carriers
  - Maximum AxC per Link
    - OBSAI: 64 max visible per link (“visible” refers to the ability to Protocol Decoded and internally use)
    - CPRI: max 124 AxC + four control streams
  - AxC Offset supported (per AxC)—Offset is programmed during AxC config
  - AxC can be deleted while link is running
  - AxC can be started/configured while link is running
- Timing and Synchronization
  - Radio Timer
    - Programmable to handle supported radio standards
    - Counts OFDM symbol boundaries
  - Physical Interface Timer—10ms frame count for OBSAI/CPRI timing
  - Timer Synchronization
    - External synchronization
      - OBSAI RP01
      - Generic strobe
    - Internal synchronization—Software (MMR) controlled synchronization
  - System Event
    - Internal System Event generation
      - Synchronize software or EDMA to Radio Timing
      - Event generation based on Radio Timer
- Data Transfers and Formats
  - Autonomous DMA
    - RAC addressing format
    - TAC addressing format
    - WCDMA delayed stream format
    - Ping/Pong L2 data format for CorePac processing
    - WiMax—Symbol format with gap at Frame End
    - LTE symbol format
    - Optional and programmable system event to mark completion of {4chip, WiMax Symbol, Frame}
  - Big and Little Endian
    - Endian Entity size programmable per DMA channel
  - IQ or QI programmable ordering
  - 16-bit I and 16-bit Q—oversampling factor per AxC 1x
  - RSA UL Data Format 8-bit I and 8-bit Q
    - Over sampling factor per AxC 2x
    - Other Over sampling factors supported via multiple AxC
    - Odd and Even Samples separated into different VBUS Quad Words
    - PHY bursts parsed to create two (odd/even) VBUS quad words
  - Generic format—Quad Word extracted from PHY bytes
- Board Topology Support
  - Point-to-Point
  - Cascade Daisy Chain
    - Link Redirect IO

- PHY
  - 8b10b line encoding / decoding
  - SerDes bit-error detection and logging
  - OBSAI: Message termination if SerDes bit error detected in header
  - Internal “loopback” test modes
  - Retransmission
    - Rule based insertion (for OFDM DL)
    - Aggregation (for WCDMA DL)
- OBSAI
  - RP3 Modulo and Dual bitmap FSM Transmission rule support
  - RP3 message reception based on address and type
  - RP3 message transmission based on transmission rules
  - RP3 message insertion
  - RP3 Circuit switched and Packet switched messages
- CPRI
  - CPRI bit interleaving and AxC Container packing/unpacking
  - CPRI Synchronization and L1 In-band protocol support
  - CPRI Support of fast and slow C&M subchannel
  - CPRI Auto RAI indication of TX link in response to LOS-or-LOF on RX Link
- Inter-device communications support
  - RP3 Packet switched messages
    - Full 6x link bandwidth support
    - Endpoint Addressing and Autonomous DMA
  - CPRI Vendor specific subchannel
  - CPRI Packet switched traffic
- Antenna data aggregation per AxC
  - 16-bit or 15-bit or 8-bit or 7-bit
  - Automatic saturation (separate I and Q saturation)
- Debug Support
  - Data tracing support
  - Clock stop and Emulation support
  - Error and Exception Interrupts (detection of many error conditions)
  - VBUS Configuration access to Data Buffer RAMs
- Frame Timing and Synchronization
  - OBSAI RP1 Support
  - Generic TRT support—MMR configured start values

Overview of AIF2 Hardware and Software Components

AIF2 is a peripheral module that supports data transfers between uplink and downlink baseband processors through a high-speed serial interface. AIF2 directly supports the following:

- WCDMA/FDD
- LTE FDD
- LTE TDD
- WiMax
- TD-SCDMA
- GSM/Edge (OBSAI only)

The antenna interface supports both OBSAI RP3 and CPRI protocols. Much of the functionality of the antenna interface is not specific to either protocol. This document does not contain detailed information about OBSAI and CPRI protocols. For more information about these protocols, see the OBSAI spec 4.0 and CPRI spec 4.1.

Antenna IQ stream data is the primary transferred data type and inter-device control data (or control data from/to the RF units) is the secondary transferred data type.

The internal AIF2 interface connects the AIF2 with the four DSP cores, RAC, TAC, and the FFTc unit via the DMA Switch Fabric and Control Switch Fabric. The AIF2 internal interface connection to the switch fabric is the VBUS. The external AIF2 interface (sometimes called the SerDes interface) connects the AIF2 with either RF units and/or other Base Band OBSAI/CPRI devices. A second internal AIF2 interface connects to the Configuration Switch Fabric for MMR read/writes and diagnostic access to the main AIF2 data buffers.

The internal interface is based on two VBUS connections to the *DMA Switch Fabric* and the *Configuration Switch Fabric*. The 32-bit VBUS connection to the 32-bit Configuration Switch Fabric is mainly for configuration, status and error/alarm condition maintenance. The 128-bit VBUS connection to the 128-bit DMA Switch Fabric is mainly used for high data rate circuit-switched and packet-switched data that is sourced from or destined to the Antenna Interface.

Both Configuration and DMA Switch Fabrics are a system of buses, switches, and internal communication protocols. The buses used by both switch fabrics are the TI proprietary VBUS and use a TI-specified VBUS protocol. The antenna interface is a slave endpoint on both the Configuration VBUS and master on the DMA Switch Fabric VBUS. Both switch fabrics use a crossbar switch (SCR) for connecting VBUS endpoints. This SCR (Switch Central Resource) allows multiple VBUS endpoints to communicate simultaneously.

AIF2 uses two basic data transport mechanisms (for data path). Data traffic is mastered by AIF2 through Multicore Navigator or Direct IO. Multicore Navigator is an internal packet DMA mechanism that is used for all other radio standard internal data transport (which is more packet-like) and non-antenna data. Direct IO is a custom state machine that transports WCDMA data to and from circular buffer in {RAC, TAC, L2, DDR3}. Circular buffering is a requirement unique to WCDMA and is not well-supported by Multicore Navigator.

The antenna external interface is implemented with six high-speed serialized links. These links support OBSAI RP3 line {2x, 4x, 8x} rates and CPRI {2x, 4x, 5x, 8x} link rates. This enables users to create a chain topology where several devices are interconnected serially.

AIF2 is intended to communicate with {FFTc, RAC, TAC, PA} subsystems without the need of constant supervision or control from CorePac software. CorePac software would initially configure the interaction between AIF2 and these subsystems; however, in steady state no MIPS are required for this data interchange.

Communications with these subsystems falls into two basic categories:

- Direct IO
  - RAC
  - TAC
- Multicore Navigator
  - FFTc
  - PA

Direct IO is used exclusively with WCDMA antenna traffic where the source and destination are circular buffers. Direct IO is similar to an EDMA solution except that the DMA is controlled by a custom FSM within AIF2. The change in approach between previous DSPs and the new KeyStone Architecture was motivated by the desire to make DMA changes dynamically (where in AIF1, the timing of these changes was difficult to achieve).

Multicore Navigator is a packet-based DMA. In LTE (for example) each LTE symbol is treated as a packet. The packets are staged in L2 RAM; The AIF2 buffers are not large enough to hold whole LTE symbols, so the LTE symbols are slowly built up in L2 RAM. AIF2 marks the LTE symbol with the received AIF2 channel number and the LTE symbol index (count value starting at 0 on the radio frame boundary). The packet is then transferred to FFTc; The whole transfer of the LTE symbol from AIF2 to L2 and finally to FFTc is performed with no interrupts of real time action by any of the CorePac cores. (packet exchanges between AIF2 and PA occur in much the same way).

Radio Standard Requirements and AIF2 Handling

Topic **Page**

3.1 WCDMA Requirements and Handling..... 41
3.2 OFDM Requirements and Handling (LTE FDD, LTE TDD, WiMax, TD-SCDMA)..... 41

3.1 WCDMA Requirements and Handling

- WCDMA AxC
  - 3.84 MHz sampling
  - Four bytes per sample
    - DL (one of the following):
      - 16-bit I + 16-bit Q, 1x over-sampling
      - 15-bit I + 15-bit Q, 1x over-sampling (CPRI only)
    - UL (one of the following):
      - 8-bit I + 8-bit Q, 2x over-sampling
      - 7-bit I + 7-bit Q, 2x over-sampling (CPRI only)
- Radio Frame Size
  - 10 ms
  - 38,400 chips (1x sample clocks)
- Data Sources/Destinations
  - RAC—Circular Buffer
  - TAC—Single Buffer, read every four chips. Treated as a circular buffer of one.
  - Memory (L2)
    - RSA processing of WCDMA data
    - Circular Buffer scheme

3.2 OFDM Requirements and Handling (LTE FDD, LTE TDD, WiMax, TD-SCDMA)

Both OBSAI and CPRI standards appear to have been developed with WCDMA sample rates in mind. The new emerging OFDM standards challenge these standards with a wide range of sampling rates. The OBSAI standard appears to be ahead of CPRI with the “Dual Bit Map” rate matching mechanism that essentially rate-matches as close as possible then inserts bubbles to make up the difference. OBSAI has also listed the values to program this mechanism to support many of the new standards configurations. CPRI does not define a mechanism, but rather the resulting packing for specific standards and frequencies. CPRI addresses only WCDMA, LTE, and WiMax.

OFDM standards differ in data processing compared with WCDMA traffic. WCDMA traffic processing is a continuous streaming operation whereas OFDM processing is chunk-based processing. In OFDM, an FFT operation is performed on AxC data immediately after UL traffic enters the device. Likewise, IFFT is performed on AxC data just before exiting the device. In both cases, the most common FFT granularity is based on the OFDM symbol (grouping of samples).

3.2.1 OFDM DMA

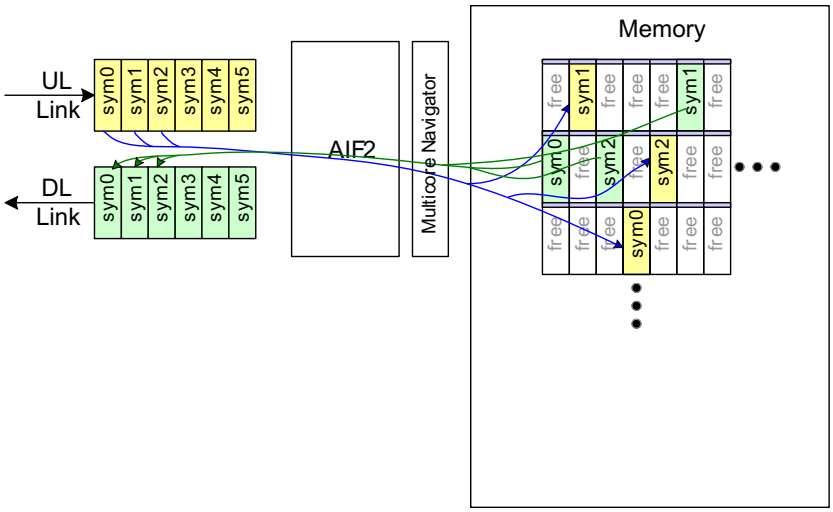
AIF2 uses the Packet DMA (PKTDMA) methodology and hardware accelerators for handling internal data movement. OFDM data is stored in memory on OFDM symbol boundaries. Multicore Navigator adds a small header (descriptor) and protocol-specific field that provides some metadata for each OFDM symbol.

Multicore Navigator is

- A DMA engine that is rate-controlled by AIF2
- A queue manager that can handle a backlog of packets to be transferred or consumed
- A memory management system

Multicore Navigator dictates (within a memory region) where to write OFDM symbols and frees memory resources after a transfer is complete.

Figure 3-1. OFDM PKTDMA



For OFDM traffic, AIF2 identifies the symbol boundaries, segmenting the traffic into symbol sized packets. Data is internally DMA'ed and stored based on OFDM symbol packet basis. The Multicore Navigator packet header (descriptor) contains such information as:

- AxC index (PS field)
- OFDM Symbol number (PS field)
- Number of valid byte in packet (Info Word)

The Multicore Navigator packet payload consists of the OFDM symbol (including cyclic prefix).

The AT (AIF2 Timer) radio timer maintains the radio standard time. Each AxC is supported to have a time offset relative to the radio timer. The AxC offset is added or subtracted to the system time calculating AxC slot boundaries and slot numbers.

The DMA buffering within the AIF2 is intentionally shallow, intended only to handle small DMA timing differences and AxC alignments. The AIF2 begins partial DMA of the OFDM packet before receiving the entire packet. The user chooses which memory the packets are delivered to. For all active AxC, AIF2 in parallel forms whole OFDM packets into user-defined memory (L2, MSMC, or DDR3).

The user configures/programs the device with the location where AIF2 will deliver packets and what should be done after the packets are fully formed in memory. The most likely next destination for a packet would be FFTC for FFT to be preformed on the packet. The Multicore Navigator subsystem can be controlled to automatically transfer (without CorePac intervention) OFDM packets (symbols) to FFTC after the packet is complete.

AIF2 and Multicore Navigator deal with OFDM by using a fixed "buffer" size. A packet does not need to entirely fill a buffer, but it must not exceed the buffer size.

For LTE in particular, there can be three different packet sizes to process, as follows:

- Extended cyclic prefix packet size
- Normal cyclic prefix packet size (first symbol of frame/half-frame)
- Normal cyclic prefix packet size (other symbols of frame/half-frame)

When configuring the Multicore Navigator buffer size, the user must choose a buffer size that can accommodate the largest-sized packet. This represents a waste of memory, but it should be an insignificant percentage of RAM as cyclic prefixes tend to be much smaller than the samples of interest.

3.2.1.1 OFDM TDD DMA {LTE TDD, WiMax, TD-SCDMA}

The OBSAI/CPRI links are full duplex (dedicated transmit and receive) and the air channel is half duplex in TDD (multiplexing between transmit and receive). The effect for the AIF2 is that approximately 50 percent of the time traffic—depending on UL/DL ratio—is idle (sending messages for which the payload is not used, zero preferred) on a given link.

3.2.2 LTE

AIF2 directly supports LTE rates as an integer multiple of WCDMA sampling. 100Mhz LTE is supported indirectly by the use of five separate 20MHz AxC (There is no special support to group, interleave, or merge the five AxC into a single entity).

Table 3-1. LTE Channels Supported

LTE Rate	Sample Rate	AxC per Link				Cyclic Prefix			FFT Size (payload)
		2x	4x	5x	8x	Normal 1st	Normal 2nd-7th	Extended 1st-6th	
1.4MHz	1.92MHz	16	32	40	64	10	9	32	128
3.0 Hz	3.84MHz	8	16	20	32	20	18	64	256
5MHz	7.68MHz	4	8	10	16	40	36	128	512
10MHz	15.36MHz	2	4	5	8	80	72	256	1024
15MHz	23.04MHz	1	2	3	4	120	108	384	1536
20MHz	30.72MHz	1	2	2	4	160	144	512	2048

There are normal cyclic prefix symbols and extended cyclic prefix symbols in LTE. AIF2 supports on-the-fly transition between the different symbol sizes by means of a control MMR. The control MMR is double-buffered, where a new value takes effect at the beginning of a frame/half-frame boundary (same timing as control of the TDD DMA control register).

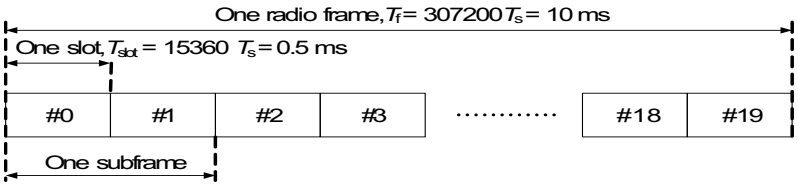
Oversampling of LTE is not directly supported, but may be achievable using the many degrees of programmability of AIF2.

- Sample bit-width of LTE is (AIF2 supported):
  - OBSAI: 16-bit I &16-bit Q
  - CPRI: 15-bit I and 15-bit Q

3.2.2.1 LTE Framing

3.2.2.1.1 LTE FDD Framing

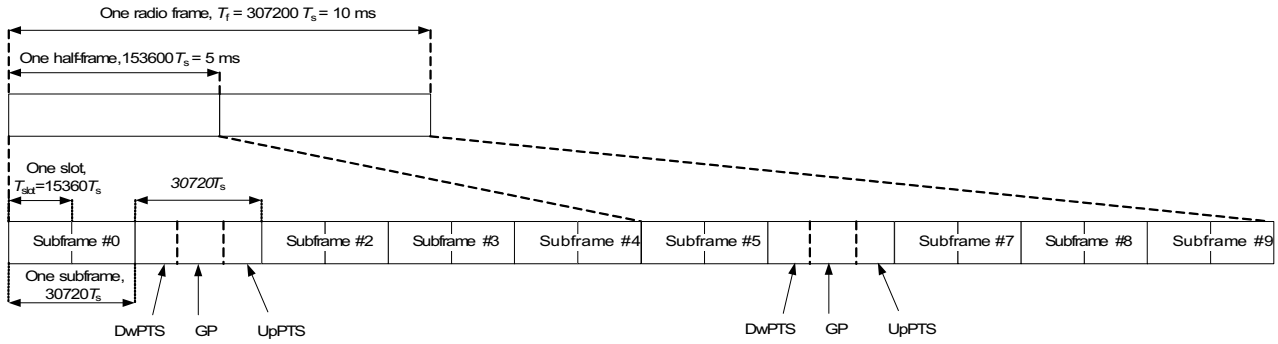
Figure 3-2. LTE FDD Frame Structure



- Framing
  - 10ms frame
  - 1ms subframe
  - 0.5 ms timeslot
  - Six or seven symbols per time-slot (that is, 20MHz LTE)
    - 307200 samples per 10ms frame
    - 15360 samples per 0.5ms timeslot
    - Extended Cyclic Prefix
      - Six symbols per timeslot
      - All same length
      - 2560 = 512 cyclic prefix + 2048
    - Normal Cyclic Prefix
      - Seven symbols per timeslot
      - 1 st symbol—2208 = 160 cyclic prefix + 2048
      - Other symbols—2192 = 144 cyclic prefix + 2048
  - Other Sampling rates
    - {0.75x, 0.5x, 0.25x} multiplication factor for sample rates of { *15MHz*, 10MHz, 5MHz}

3.2.2.1.2 LTE TDD Framing

Figure 3-3. LTE TDD Frame Structure (5 ms Switch Point Illustrated)



- Framing
  - (same as FDD case)
  - Plus… half frame 5ms concept
- LTE cyclic prefix is static and can not be modified while a link is active.
- Dynamic change of UL/DL ratio—Switch Point: 5 ms half frame boundary or 10 ms frame boundary
- Special LTE TDD subframe
  - No implication for AIF2
  - Symbols are simply used for different uses
  - Symbols are same size as non-special timeslots
  - Two options for Switch Point
    - Case1 (5ms) one special frame per 5ms
    - Case2 (10ms) one special frame per 10ms
    - AIF2 needs to support change in UL/DL ration on 5ms or 10ms basis
- TDD Muxing
  - Each timeslot can be either UL or DL
  - Ratio can be changed at the switch point.

AIF2 does not use the subframe or slot concept. It simply counts n symbols within a frame/half-frame. An egress and ingress channel is allocated for each AxC. A bit map per channel indicates on a symbol-by-symbol basis whether the symbol is to be DMA'ed or whether the symbol is empty. It is either the CSL or programmer's responsibility to remap the LTE TDD UL/DL partition into the appropriate DMA control bit maps. For PD, tdd_en_n field in PD DMA channel config registers could be used. This 144 bits correspond to each symbol and these bit maps work per bit to support several radio standards. For LTE, there are 120 or 140 symbols in one frame for UL or DL; the user can only set each bit to one or zero for their purpose. On every frame or half-frame boundary, use of the bit map will rewind to the beginning. The bit map may be altered on-the-fly to change the UL/DL ratio.

PE does not have a bit map. Instead, it has a one-bit tdd_axc field in the PE DMA channel 0 register to turn on/off TDD mode for egress. If it is turned on, PE inserts a symbol only when it receives the packet from DB FIFO. If user doesn't push or transfer a packet for some symbol time, PE considers it as a UL symbol (not activated symbol) time and inserts zero data automatically in those slots.

AIF2 also does not use special frames. The user programs the DMA bit maps so the DL symbols are handled by the egress channel and UL symbols are handled by the ingress channel. This is how AIF2 generalizes traffic so it can handle multiple radio standards with a single hardware interface.

3.2.2.2 CPRI LTE (FDD and TDD)

3.2.2.2.1 CPRI LTE (FDD and TDD) Packing of AxC

Only the following LTE rates are directly tested in AIF2: 5MHz, 10MHz, and 20MHz (TI believes these are the main frequencies of interest). These LTE rates have an actual sampling rate of 7.68MHz, 15.36MHz, and 30.72MHz sampling rates that correspond to 2x, 4x, and 8x of WCDMA sampling rates.

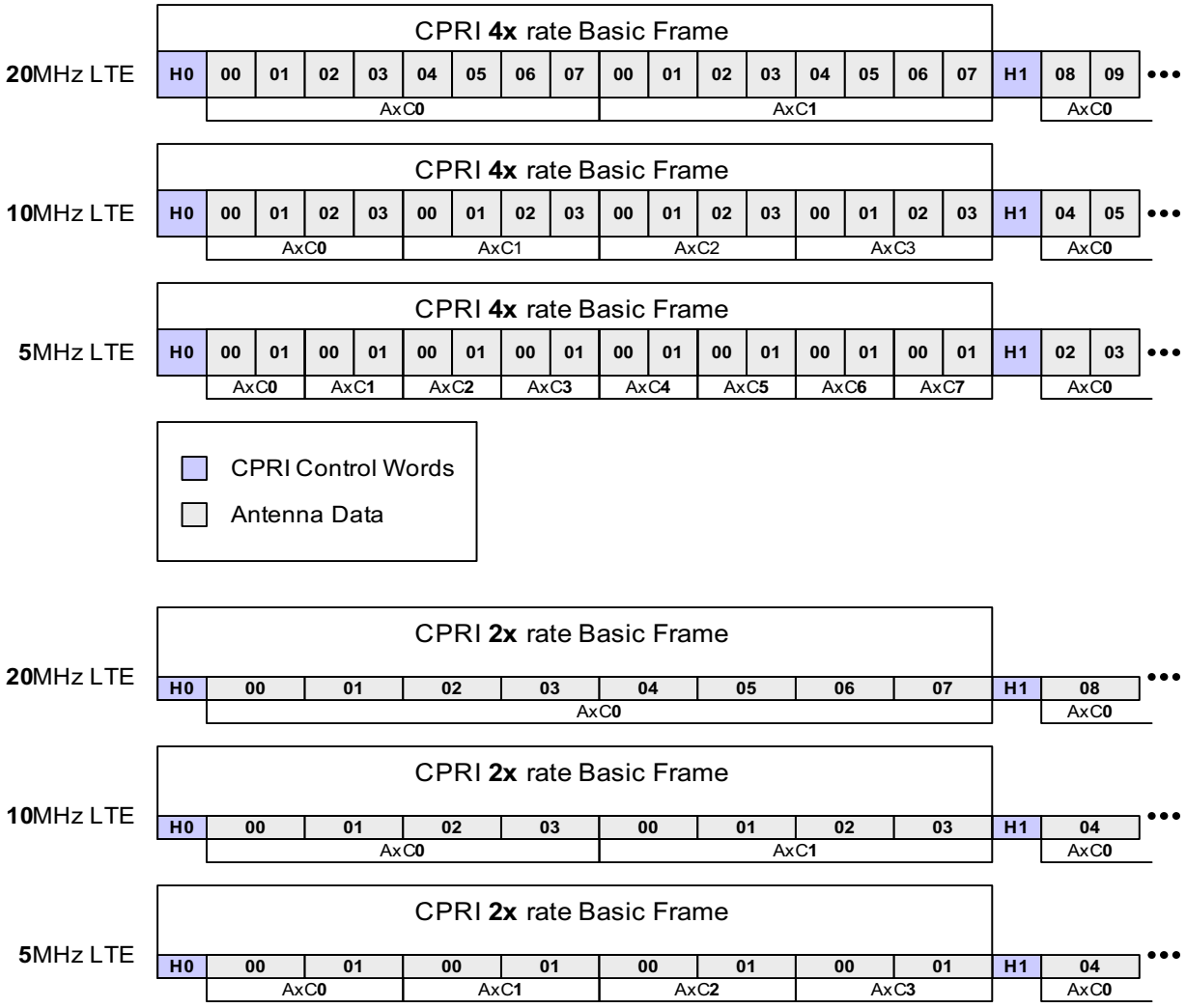
AIF2 supports only CPRI LTE sample bit widths of 15-bit I + 15-bit Q. At these sampling rates, the following number of LTE AxC is supported at the different rates.

Table 3-2. LTE AxC Supported Over Different CPRI Link Rates

	Link Rate				
LTE	**1x**	**2x**	**4x**	**5x**	**8x**
5HMz	2	4	8	10	16
10MHz	1	2	4	5	8
20MHz	0	1	2	2	4

LTE is a natural multiple of the WCDMA rate where the LTE rates {20MHz, 10MHz, 5Mhz} rates correspond to the exact integer multiple of {8x, 4x, 2x} the WCDMA antenna rate. The packing is chosen as if the LTE data were oversampled WCDMA data. That is, the 20MHz LTE case for a 4x rate link has eight samples of antenna 0 (AxC0) followed by eight samples of AxC1.

Figure 3-4. CPRI LTE Sample Packing {20MHz, 10MHz, 5MHz} vs. {4x, 2x} Link Rate



3.2.2.3 OBSAI LTE (FDD and TDD)

With 16-bit I + 16-bit Q sample sizes, exactly four samples fit into an OBSAI message; each OBSAI message contains four samples for one AxC.

The five-bit OBSAI timestamp starts at zero (on a radio frame boundary) and increments by one every four samples for a given AxC.

The AT (AIF2 Timer) is used to predict LTE (and other standards) frame boundary that is used in timestamp Tx generation and Rx checking.

OBSAI RP3 has done a really nice job of listing all the LTE channel bandwidths and listed the programming of the dual-bit-map FSM mechanism for rate matching.

Please see the OBSAI RP3 specification, Appendix F for details.

While the OBSAI RP3 standard does not specially call out LTE TDD, it is assumed that zero samples continue to flow through the interface while a UL or DL link is idle. AIF2 expects this null data (null data in the sense that the data is not used) and continues checking the timestamp of these null messages.

3.2.3 LTE 80 MHz and 100MHz Support

The LTE standards body has not fully developed the concept of 100MHz LTE. There are two possibilities:

- 80MHz: four times the bandwidth of 20MHz LTE
- 100MHz: five times the bandwidth of 20MHz LTE

3.2.3.1 80 MHz

Four times the bandwidth of 20 MHz LTE does fit into a single 8x rate AIF2 link. AIF2 does support modulo transmission rule (which is the way to program OBSAI for this AxC traffic rate).

3.2.3.2 100 MHz

A single 100MHz AxC will not fit in the highest supported link rate (8x). At best, four 20MHz LTE AxC would fit into a single 8x rate link. A single 100MHz LTE AxC has the same bandwidth as five 20 MHz AxC.

One possible method of supporting 100 MHz LTE is for the sending device to segment the AxC into five parts, then send these parts over five different 20 MHz LTE AxC. If it is done correctly, the Multicore Navigator mechanism could be manipulated on the receiving device such that only Multicore Navigator headers would need to be modified in order to reconstruct the 100MHz AxC from the five 20 MHz AxC.

To facilitate this reconstruction, the AIF2 PKTDMA channels involved should be programmed to use Multicore Navigator “Host Mode” instead of Multicore Navigator “Monolithic Mode”. Host mode separates the Multicore Navigator descriptor from the Multicore Navigator buffer. Host mode also allows for multiple descriptor/buffer pairs to be linked-listed into a larger packet structure. In the most optimal usage, the Multicore Navigator buffers should be sized such that a single LTE 20 MHz symbol fits into a buffer. In this way, there will be a single Multicore Navigator descriptor/buffer pair per 20 MHz Symbol. After all 20 MHz symbols are received, the application software can manipulate the buffer pointers linking all five descriptor/buffer pairs into a single LTE 100 MHz symbol packet.

3.2.4 TD-SCDMA

AIF2 CPRI TD-SCDMA support is a comprehensive solution to the TD-SCDMA market. AIF2, in conjunction with Multicore Navigator, seamlessly transports and coordinates TD-SCDMA traffic with application software running on the CorePac core.

For TD-SCDMA, the DMA granularity may need to support two different modes:

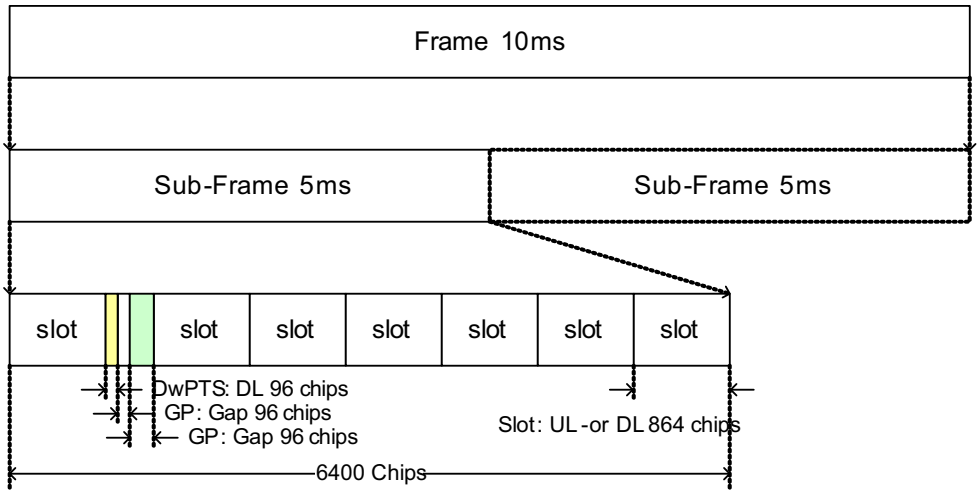
- DMA on Symbol Boundaries
- DMA on ½ Symbol Boundaries

½ Symbol Boundary DMA is not supportable with OBSAI

3.2.4.1 TD-SCDMA Framing

TD-SCDMA is a TDD standard meaning that the Air channel is multiplexed between UL and DL.

Figure 3-5. TD-SCDMA Frame Structure



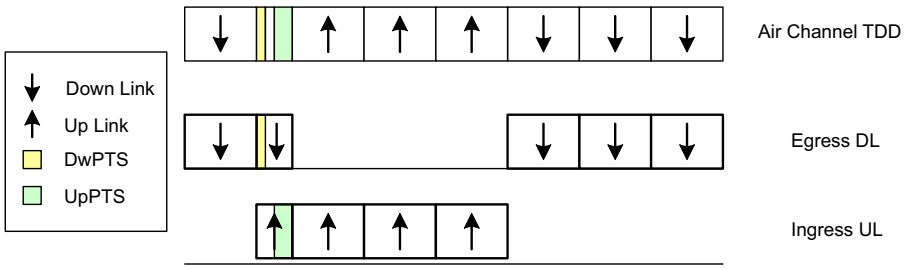
TD-SCDMA has a 10 ms frame that is broken down into two 5 ms subframes. Each subframe is further broken down into

- Seven slots for transporting antenna traffic—Each are 864 chips
- Three special slots:
  - DwPTS: DL 96 chips
  - GP: Gap (unused) 96 chips
  - UpPTS: UL 160 chips

The first of the seven slots is always DL data. The special slots always follow the first normal slot in the subframe. The remaining regular slots can be configured to be either UL or DL information with the following restrictions:

- If there are any UL slots, the first UL slot follows the special slots
- All UL slots are grouped together with only one transition to DL slots
- All remaining DL slots are grouped together

Figure 3-6. TD-SCDMA Example DL/UL Mix



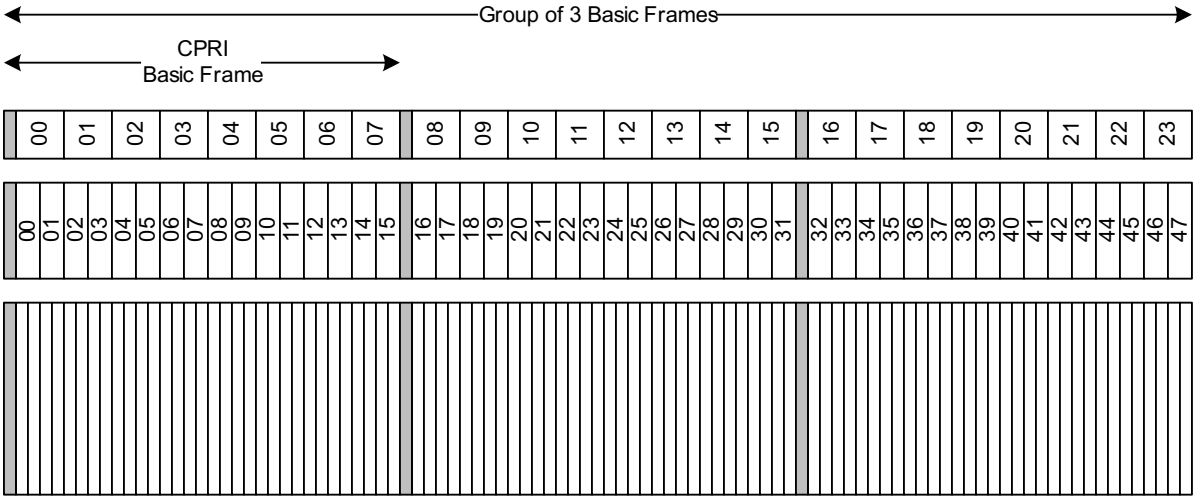
The air channel is TDD (simplex), but the CPRI SerDes are not simplex; the SerDes are full duplex. The SerDes have two half links (Tx and Rx) with DL traffic on one half (that is, TX) and the UL traffic on the other half (that is, RX). This has the unavoidable impact that only one half of the SerDes bandwidth is usable in TDD systems.

There are three separate “special slots” in TD-SCDMA, but for simplification, AIF2 treats all three special slots as grouped together or a single slot. The next special handling aspect of AIF2 is that the grouped special slot is handled redundantly for both UL and DL. This special handling is transparent at the RF card as the CorePac fills in zeros for all unused portions in the DL case and CorePac disregards unused portions in the UL case.

The AIF2 does not perform any special processing for 2x oversampling. The user would need to allocate two separate antenna carrier bins where AIF2 would simply treat Odd and Even samples as separate antennas. If the user requires odd and even samples interleaved, the CorePac would be required to perform this function.

3.2.4.2 CPRI TD-SCDMA

Figure 3-7. TD-SCDMA AxCs Packed into Three CPRI Basic Frames



CPRI was developed initially for WCDMA and because TD-SCDMA is exactly one-third the sample rate, TD-SCDMA fits into the CPRI framing. For CPRI WCDMA (15-bit mode), a sample for each of {8, 16, 32} AxC fits respectively in a {2x, 4x, 8x} link rate basic frames; the same is true for TD-SCDMA. The difference between WCDMA and TD-SCDMA is:

- WCDMA: The next basic frame contains the next sample for those AxC
- TD-SCDMA: The next two basic frames contain a sample for other AxC… Effectively 3x more AxC are supported.

The CPRI link rates {2x, 4x, 8x} support {24, 48, 96} AxC.

The AIF2 supports a maximum of 128 ingress DMA channels, effectively limiting the maximum “visible” supported TD-SCDMA AxC per device to 128. (The same is true for egress).

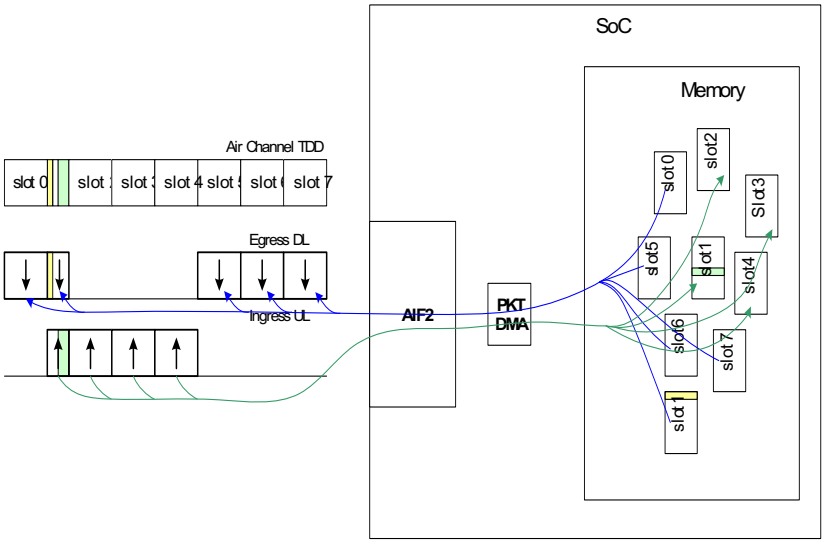
There are a total of 38,400 basic frames per 10 ms frame. For TD-SCDMA support, these basic frames are grouped into threes giving a total of 12,800 basic frame groups per 10 ms frame.

An essential function of AIF2 is to pack and unpack the CPRI frames with AxC data. Supporting this function, AIF2 has 128x ingress and 128x egress FIFOs to support this pack/unpack operation. The user can configure which AIF2 FIFO is mapped to which CPRI TD-SCDMA antenna slot.

3.2.4.3 TD-SCDMA DMA

The DMA and memory storage of TD-SCDMA Data is slot-by-slot based. Each slot of TD-SCDMA data is stored in a contiguous memory block known as a buffer. While AIF2 is receiving TD-SCDMA AxC data, it counts through the samples and identifies slot boundaries. AIF2 tightly coordinates with the PKTDMA subsystem to transport the slot into memory.

Figure 3-8. TD-SCDMA Slots Transferred To/From Memory



AIF2 supports AxC offsets. Each AxC can have different frame alignment than the CPRI link and each other. The AxC offset is a positive value programmed on an AxC-by-AxC basis which represents a number of sample offsets relative to the beginning of the CPRI frame.

Once the slot is completed in memory, the Multicore Navigator queue manager presents the buffer to the CorePac core. A pointer to the buffer is placed into a queue that is accessible to the CorePac core. The application software has great flexibility in configuring and use of these buffers. A system event can be programmed to alert the CorePac that an arrival queue is non-empty; the CorePac core can use this system event either as an interrupt or simply poll the event. The application software can consume the data very quickly or can allow multiple slots of data to accumulate in the arrival queues. A single arrival queue may be used for all AxC, or multiple arrival queues may be used to facilitate some form of priority scheme.

The separation of UL and DL has several degrees of freedom. Because UL and DL are separated by ingress and egress links, the AIF2 and Multicore Navigator can be programmed to place UL and DL traffic in separate memory regions such as different L2 regions for different CorePac cores. The mixture between UL and DL traffic is also highly programmable. Even though TD-SCDMA only allows for two transition points (switch points) between UL and DL traffic within a subframe, the AIF2 is highly programmable and has no such limitation. AIF2 allows the allocation of UL and DL slots to be programmed on-the-fly. From one TD-SCDMA frame to the next, the slots can be reallocated between UL and DL.

The AIF2 bit map per DMA channel is programmed with indication bits configuring AIF2 for:

- 1’b1: slot is active and DMA data used
- 1’b0: slot is null, no internal DMA for slot
  - Tx: zeros filled into CPRI link
  - Rx: data is not extracted from CPRI link

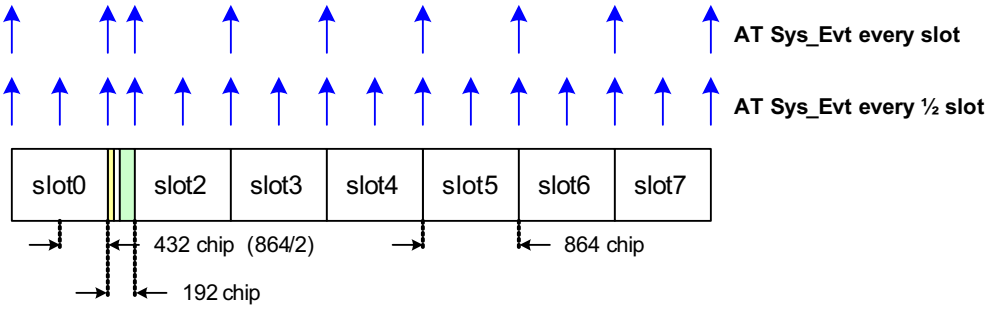
The example tdd_en 16-bit field setup in PD register looks like below.

PD Ingress… bit0=0, 1, 1, 1, 1, 0, 0, 0, x, x, x, x, x, x, x, bit15=x

PE does not have this bit field register and only has a tdd_axc enable/disable field in the PE link register. Users control the TDD symbol transfer by not transferring symbol data if symbol slot is for UL (not active). The user also may use protocol specific field in monolithic descriptor to know the symbol sequential number and AxC number.

3.2.4.4 TD-SCMDA Timer (AT) Operation

Figure 3-9. TD-SCDMA (AT) Timer-Based System Event Generation

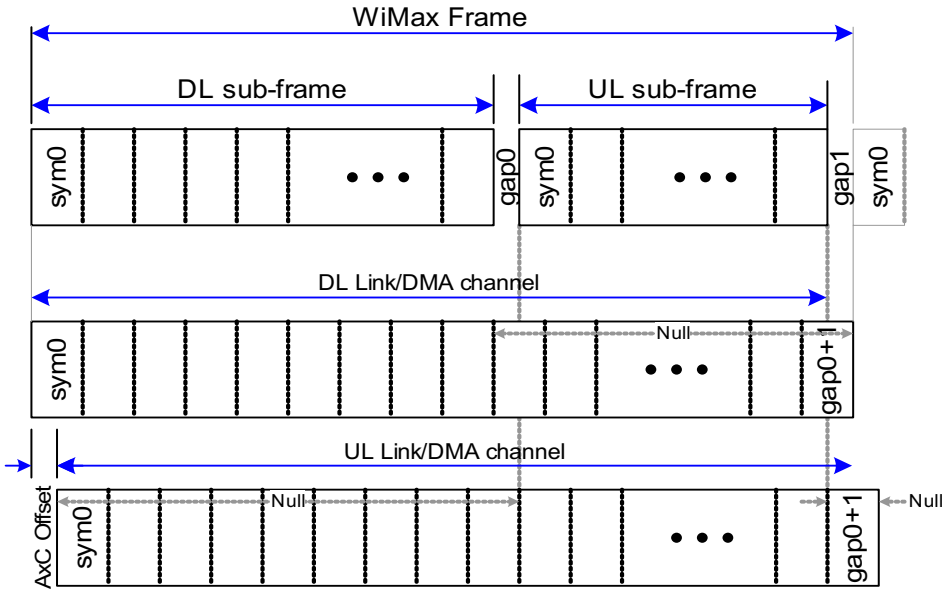


The AIF2 timer (AT) can generate system events for use by application software running on the CorePac cores. The AT is highly programmable. As a minimum requirement, it can supply timing strobes on TD-SCDMA slot or one-half slot boundaries by using modulus count register. A system event is generated at the beginning of the special region and at the end of the special region, but system events are not generated on special slot boundaries within the “special” region.

3.2.5 WiMax

3.2.5.1 WiMax Framing

Figure 3-10. WiMax Frame Structure



- OFDM samples n= {128, 512,1024,2048}
- cyclic prefix p= {n/4, n/8, n/16, n/32}
- Symbol size n + p
- sample rate 28/25 x {1.25MHz(128), 5MHz(512), 10MHz(1024), 20MHz(2048)}
  - 7MHz and 8.75MHz also have importance
- Over-sample 1x
- Data width 16-bit I and 16-bit Q

The WiMax TDD frame is partitioned into two subframes with a gap in between:

- DL subframe
- UL subframe

The DL and UL subframes are always an integer number of WiMax symbols in length. All symbols are the same length in WiMax and comprise an FFT length of samples plus a fixed-length cyclic prefix (combination of these two is referred to as a symbol in this document). The frame length MOD symbol length is the gap length (always smaller than a symbol in length).

On a frame-by-frame basis, the DL portion can shrink or grow by an integer number or blocks with the UL portion growing or shrinking by the amount. The gap between UL and DL does not change as the UL and DL portions are adjusted in duration.

3.2.5.2 WiMax, AIF2 Implementation

The AIF2 generic mechanism is used to implement WiMax. As with all TDD processing, AIF2 uses two separate DMA channels to implement a single WiMax TDD antenna:

- UL: Ingress DMA channel
- DL: Egress DMA channel

Both channels are configured with n+1 symbols of length where:

- First n symbols programmed as symbol length
- Last symbol programmed as gap length (combined length of two gaps)

AIF2 has the generic concept of an AxC offset that allows antenna carriers to be offset relative to the system timer (and each other). For WiMax, the offset mechanism is manipulated to create the subframe time offset between DL and UL. The DL AxC offset is programmed to be what ever the user requires. The UL Offset is programmed as the DL offset plus the time duration of the first gap; this is just to align the UL symbol boundaries.

Just as with all other TDD standards, the PD AIF2 bit map per DMA channel is programmed with indication bits configuring AIF2 for:

- 1’b1: symbol is active and DMA data used
- 1’b0: symbol is null, no internal DMA for symbol
  - Tx: zeros filled into OBSAI/CPRI link
  - Rx: data is not extracted from OBSAI/CPRI link

AIF2 PD has additional four DMA channel config registers (total for 144 symbols) to support tdd_enable set for more than 16 symbol case.

3.2.5.3 OBSAI WiMax

OBSAI RP3 lists all the WiMAx channel bandwidths and the programming of the dual-bit-map FSM mechanism for rate matching.

See the OBSAI RP3 specification, Appendix D for details.

3.2.5.4 CPRI WiMax

AIF2 has adopted the Dual Bit Map FSM (DBMF) concept from OBSAI for purposes of CPRI rate-matching operation. In CPRI mode, the operation works a little different than in OBSAI Mode.

The granularity of packing operation in CPRI mode is the sample 32-bit or 30-bit sample. It is disabled for WCDMA.

- User defines X AxC to be burst transferred
- User defines M bubbles to be inserted (when indicated) between bursts of AxC
- (at the end of each burst of X samples) DBMF algorithm indicates when to insert a burst of M bubbles, or simply to progress to the next burst of X samples.

3.2.6 GSM/Edge Requirement and Handling

- Ping/pong buffer (buffer per timeslot)
- Each timeslot
  - Uniform timeslot buffer size… responsibility of RF card
- GSM Timing
  - GSM frame is 60ms/13
  - GSM timeslot (60ms/13)/8
  - Need 3.5 hour timer… 26 x 51 x 2048 x frames
    - Need unique frame number in 3.5 hour timer
  - GSM rate is not even multiple of RP1 30.72MHz clocks
    - AT needs to approximate timing of GSM to nearest 30.72MHz clock

System events are required on GSM timeslot boundaries and 1/64 th GSM timeslot boundaries. The GSM sample clock is not in a nice multiple of the OBSAI byte clock rate, so an approximation is performed. It is observed that every 13 GSM frames, the GSM timer is an exact multiple of byte clock which occurs every 60 ms.

- 18432000 307.2MHz clocks every 60 ms
- 104 GSM timeslots per 60ms
  - 24 timeslots w/ 177230 clocks
  - 80 timeslot w/ 177231 clocks

Table 3-3. GSM Slot 307.2 MHz Approximations

	307.2MHz clocks per GSM slot (Approximations)							
	Slot0	**Slot1**	**Slot2**	**Slot3**	**Slot4**	**Slot5**	**Slot6**	**Slot7**
Frame0	177231	177231	177231	177230	177231	177231	177231	177231
Frame1	177230	177231	177231	177231	177230	177231	177231	177231
Frame2	177230	177231	177231	177231	177231	177230	177231	177231
Frame3	177231	177230	177231	177231	177231	177230	177231	177231
Frame4	177231	177231	177230	177231	177231	177231	177230	177231
Frame5	177231	177231	177230	177231	177231	177231	177231	177230
Frame6	177231	177231	177231	177230	177231	177231	177231	177230
Frame7	177231	177231	177231	177231	177230	177231	177231	177231
Frame8	177230	177231	177231	177231	177230	177231	177231	177231
Frame9	177231	177230	177231	177231	177231	177230	177231	177231
Frame10	177231	177230	177231	177231	177231	177231	177230	177231
Frame11	177231	177231	177230	177231	177231	177231	177230	177231
Frame12	177231	177231	177231	177230	177231	177231	177231	177230

- GSM subslot System Events
  - 64 system events per slot
  - 2769 307.2Mhz clocks per segment
  - Last segment of timeslot is either 14 or 15 clocks longer than the 2769 clock segments

3.2.6.1 GSM Base Band Hopping

- DMA
  - Have a fixed PKTDMA channel per all possible AxC
  - App software chooses which DMA channel to write to
  - Have a single PKTDMA channel for all Egress GSM control
    - Use Protocol Specific fields in Multicore Navigator header to pass OBSAI address from APP software to AIF2
- Data
  - Both UL and DL data has a (DTX) non-transmission portion at the end of each timeslot
  - DL has two forms, Compressed (mostly DTX) and normal (only a few DTX messages)
- Tx Rules
  - Use fixed DBM rules, mapping all possible antenna carriers
  - Fixed OBSAI adr and type per AxC (for AxC traffic)
  - Use OBSAI adr from packet (for GSM Control)
- Hopping
  - If PKTDMA does not send a packet in time for transmission on a given channel, then empty messages are sent in place of real data
- Timestamp
  - MSB is 1 for beginning of timeslot (0 else)
  - Five LSBs count up normally, wrap every 32
  - DL, first message MSB is active first message
  - UL, first four messages, MSB is “1”
- Error conditions
  - If last message of timeslot is corrupted, need watch dog or other timer to close out EOP
  - If first message of UL is corrupted/lost, cannot lose whole packet

Normally, one device will process a set number of fixed antennas of data. In baseband hopping, this antenna contribution is dynamically mapped to different physical antennas on the RF card.

The basic radio concept is that multiple GSM frequency bands are supported by a single radio tower. Some of these frequencies have lower noise floors than others. With Baseband hopping, the “good” and “not-as-good” GSM frequency bands are TDM shared by all the active antenna traffic.

The implications on AIF2 transmission are:

- The mapping of PKTDMA streams can be dynamically reassigned on GSM timeslot boundaries. All possible transmission rules are predefined at configuration time (even though only some of these will be active at any point in time)
- The App software will dynamically map which antenna to transmit on simply by targeting a different Multicore Navigator queue. AIF2 is tolerant of data being present (or not) on a timeslot by timeslot basis.
- For GSM control data, Multicore Navigator passes the OBSAI address to allow for the OBSAI address to dynamically remap. App software writes the OBSAI address fields into a Multicore Navigator field. The PE will use this OBSAI address when building messages (not using the default OBSAI address programmed for the channel).
- AIF2 PE is tolerant of whole missing GSM Multicore Navigator packets. The PE will expect that data may “hop” from one PKTDMA channel to another and tolerate these gaps.

- AIF2 PD is also tolerant of whole missed GSM timeslots.

Example:

If the baseband pool is supporting 36 AxC, each AIF2 in each device will have 36 identically mapped transmission rules. If each device is supporting only eight of the 36 AxC, the application software will write data to eight of the 36 Multicore Navigator queues with GSM data for each GSM timeslot. The AIF2 will process only those channels that have data for that particular timeslot.

For each timeslot of egress GSM data, the application software also supplies control data associated with the RF destination of the AxC data. The application software controls or routes the transfer of the control packets by writing the appropriate OBSAI address into a protocol-specific word attached to the Multicore Navigator header. AIF2 inserts this software-written information into the OBSAI header when constructing the OBSAI message.

BB hopping has the specific requirement that a single device can dynamically target different RF destinations with control information. AIF2 has a very specific feature that allows this behavior; application software can dictate the OBSAI address in the header of the control message (when the message is in L2). When using this mechanism, a single OBSAI transmission rule can be used for many different control messages with different destination addresses.

In most cases, GSM samples are not 16-bit or 8-bit. AIF2 should be programmed to treat GSM data as generic 32-bit data in these cases. GSM Timestamping is not frame-based, but rather is timeslot-based. This enables whole timeslots to be omitted without regard for maintaining gaps in timestamp generation/checking. It is highly advisable that all GSM channels/AxC within a hopping group have the same AxC offset. The AIF2 would probably work with different AxC Offset, but this makes no sense from a system perspective.

3.2.6.2 AxC Compression (Time Slot DTX)

AIF2 is required to handle both compressed and non-compressed GSM. The concept of compression is simply that some portion of the end of a Time Slot is not populated with antenna data.

The implications on transmit are that the PE will not fail the link if the DMA data does not fully fill the GSM Time Slot. The PE will transition to empty messages (OBSAI) or zero antenna data (CPRI) in this event. PE will commence with the next packet at the beginning of the next GSM Time Slot.

The implications on reception are nearly identical. The PD is tolerant of streaming data which does not quite fill the GSM Time Slot and will expect Timestamp to restart at the beginning of the next GSM Time Slot.

3.2.6.3 Dynamic Configuration

- Link Add/Delete
- AxC Add/Delete
- GSM Baseband Hopping
- LTE (TDD) and WiMax (TDD)
  - Change UL/DL ratio (on-the-fly)
- Timing
  - System Event Add/Delete
  - RadT resync (on-the-fly)

AIF2 supports several forms of dynamic changes that support changes to the basestation radio configuration. These types of changes could be split into two basic categories:

- On-the-fly: Change occurs from one frame to the next without any OFF or error period. Requires special ping/pong buffering of configuration data where the ping/pong will toggle on the next frame boundary.
- Normal Changes: An antenna carrier is torn down, later rebuilt to accomplish a change. There is the expectation that some small amount of time of no transmission will elapse.

Basic operation of links and antenna carriers allow Add/Delete support. On-the-fly modification is generally not supported. To perform a Modify operation, delete the link/AxC followed by an Add with the new characteristics.

When Adding or Deleting a channel, there are three levels of configuration required:

- PHY: Links need to be enabled and preferably up and running
- Protocol: OBSAI address, type, transmission rules setup
- DMA: DMA channels assigned and turned on

WCDMA AxC Add/Delete should incorporate the Add/Delete of the assigned DirectIO DMA configuration. The DirectIO engine does not ping/pong buffer channel configuration, but does time the application of configuration data to the RadT frame boundary (indicated by an internal system event sourced from AT). With PKTDMA, the user has the choice to Add/Delete Multicore Navigator channels when performing Add/Delete of AxC, or simply keep the Multicore Navigator channel active always (as no data will transfer when the AxC is OFF).

The GSM Base Band hopping special requirements are not handled as reconfiguration at all, but rather by allowing software to direct where data goes. For antenna data, all DMA channels are allocated, and software steers data into the appropriate DMA channel FIFO. For control data, software writes the destination OBSAI addressing into the Multicore Navigator packet.

The standards that support TDD have separate UL and DL regions. On-the-fly update of the DL/UL ratio is required. On-the-fly update requires AIF2 to ping-pong buffer control information and transition to new control information at a precise timing (that is, on frame boundary).

Table 3-4. OFDM On-The-Fly Update Requirements

Radio Standard	On-the-fly Update		
	UL/DL Ratio	Symbol Size	Sample Freq or Bandwidth
LTE FDD	N/A	No	No
LTE TDD	Yes	N/A	No
WiMAX TDD	Yes	No	No
TD-SCDMA	Yes	N/A	No

LTE supports extended and normal cyclic prefix length in symbols and multiple bandwidth. It is expected that some users will support multiple LTE configurations on the same device (and even on the same link). LTE bandwidth change (for example from 20 MHz to 10 MHz) or change of cyclic prefix length is a radical reconfiguration of an antenna carrier. For this change, the antenna carrier is deleted, later added with the new parameters. This is not considered on-the-fly because there a short period of down time between (10 ms).

The user has a choice of handling Multicore Navigator when supporting multiple LTE bandwidths simultaneously. The easiest (and most practical) solution is to allocate the Multicore Navigator buffer size to accommodate the largest symbol size supported. When an antenna of lesser bandwidth uses these buffers, only a fraction of the memory is actually utilized. When using this simple approach to Multicore Navigator memory management, there is no Multicore Navigator reconfiguration required when changing LTE bandwidth.

Changes of system event generation are supported by first turning a system event off, reconfiguring the system event, then turning it on. As a rule, system events are mainly used by application software and that having an OFF period of time is required for software to resync to the difference in system event timing. Each system event is independent of the others, giving the flexibility to change one without corrupting any others.

The AT Radio Timer (RadT) is synchronized at startup to an external timing source. Also supported is the capability to resynchronize the RadT dynamically. In this operation, RadT synchronizes to the next synchronization point (frame boundary).

Dynamic AxC channel reconfiguration example (LTE case)

Egress Channel

- Stop pushing packets into the Tx queue
- Allow channel to starve by waiting some time
- Disable EDB if needed
- Disable the AxC channel at PE
- Wait for at least 1 radio frame of time
- Reconfigure PE channel
- Reconfigure EDB channel if needed
- Enable EDB if it was disabled
- Enable PE channel
- Start pushing Tx data about 1 ms before the next frame boundary (14 descriptors)

Ingress Channel

- Disable PD channelWait for at least 2 radio frame of time
- Disable IDB and PkeDMA if needed
- Reconfigure PD channel setup or common setup
- Reconfigure IDB channel or Rxflow if needed
- Enable PktDMA and IDB if it was disabled
- Enable PD channel

User can disable/enable IDB for changing the FIFO size but it should be done after the PD channel is disabled. Disabling or enabling PktDMA for Rxflow reconfiguration is not a must. It could be safely done in run time. BW change requires Channel LUT value change and this should be done only when all related AxC channels are disabled.

AIF2 Timing Information

Topic Page

4.1 Timing and Topology 61
4.2 AxC Offset 63
4.3 OBSAI Reception Timing 65
4.4 PhyT and RadT Timers 70
4.5 AIF2 Timing Details 71

4.1 Timing and Topology

Within the daisy chain, the user needs to define the timing reference. One logical choice is to define time equal to zero at the radio head. This would yield all DL timing to be negative time and all uplink timing to be positive time. An alternative is to define time=0 at the first DL node in the chain; this way, all offsets are positive.

With any approach, each device in the chain has some amount of propagation delay between its input and output links. This translates into approximately 228 ns (around WCDMA one chip) of timing difference between adjacent devices in the chain. AIF2 has sufficient buffering capability in most topology cases.

Figure 4-1. Timing Topology, Multiple RF, Common Alignment

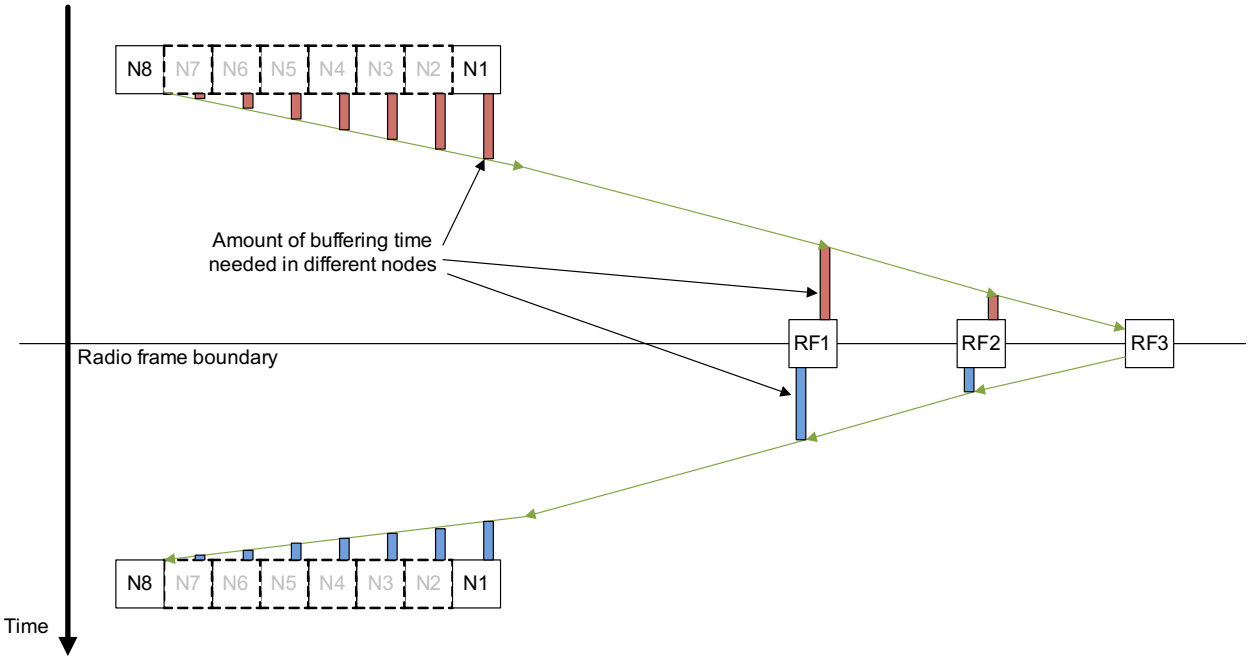
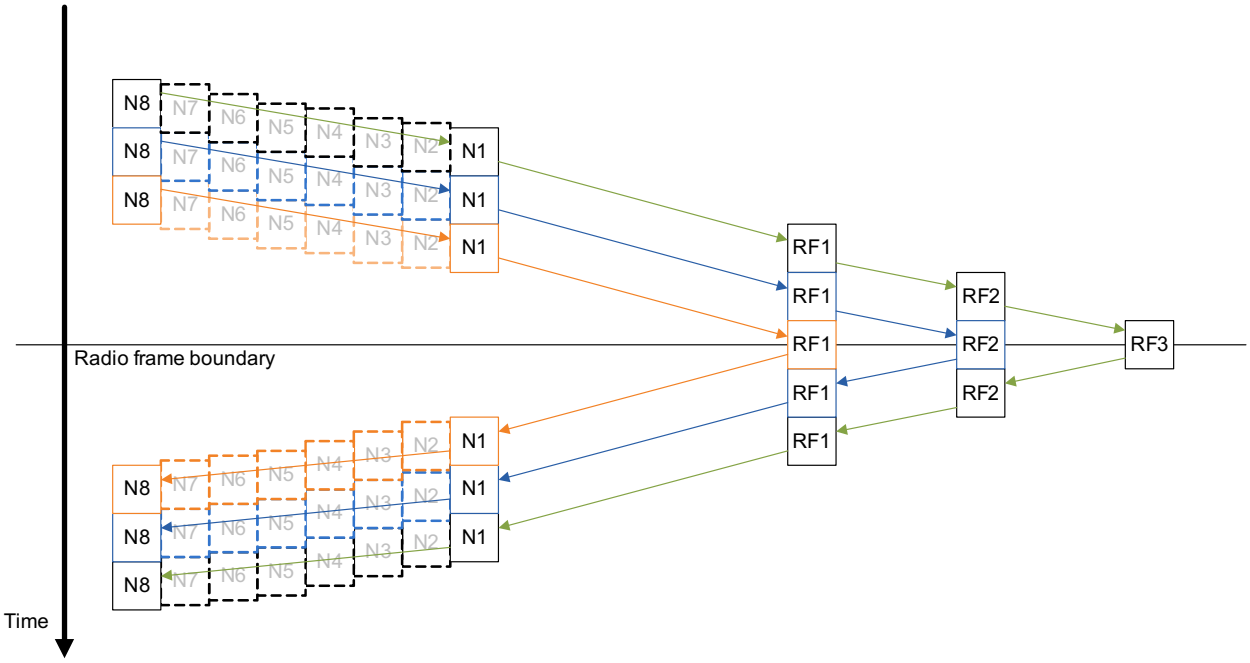


Figure 4-1 illustrates that each node in the chain has separate radio timing alignment. Node RF3 is defined with radio time equal to zero, where DL offsets are negative (red) and UL offsets are positive (blue). The offset differs at each point in the chain correlating to the propagation delay through the devices. In this example, there are three RF units in the serial chain. RF1 and RF2 both are implementing offsets because they are not defined to be equal to zero timing references.

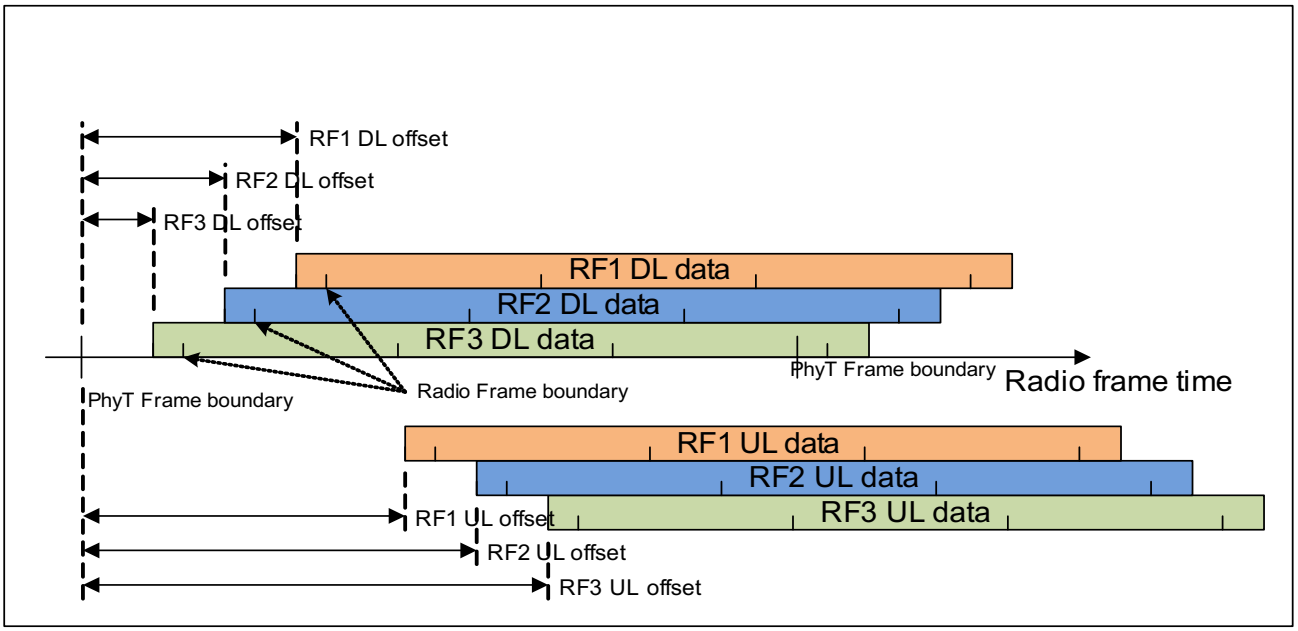
In topologies with multiple chained RF radio heads, the definition of time equal to zero can be further confused. Optionally, the furthest radio head from the baseband card can be defined with time equal to zero. In this case, the closer radio heads would delay DL data locally such that all radio heads would have synchronized transmission alignment. The radio heads would also result in synchronized reception timing, where closer RF heads would delay UL signals, compensating for any fiber delay to the furthest radio head.

Figure 4-2. Timing Topology, Multiple RF, Independent Alignment



Optionally, each RF head could have its own unique concept of time equal to zero relative to itself. In this case, it is envisioned that radio head traffic is separated by link, where each link has a different reference point for timing. This is a good example to show why each AIF2 link supports independent timing.

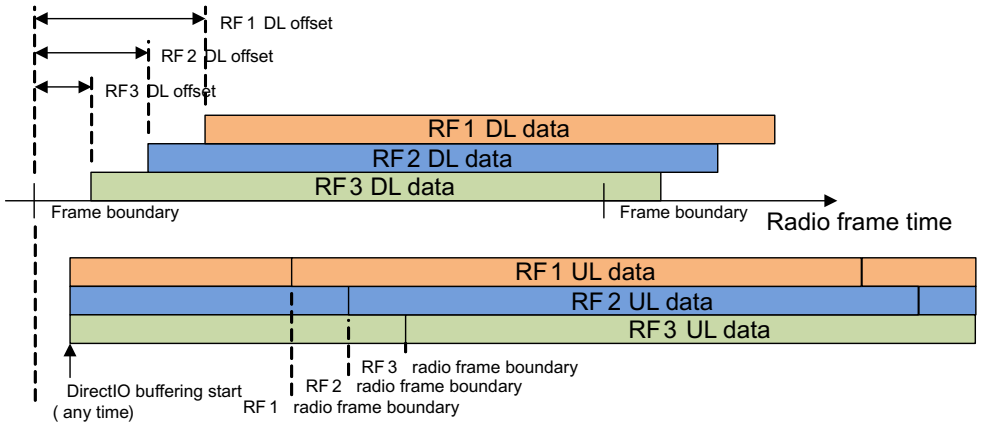
Figure 4-3. OFDM/GSM/Edge2 Timing



For all radio standards, AIF2 carefully predicts OBSAI timestamping. On reception, the timestamping is used to begin radio frame detection and then continually check the inbound alignment. On transmission, the timestamp is generated and appended to the outgoing radio traffic.

For all radio standards other then WCDMA, DMA is aligned to radio time. For {LTE, WiMax, TD-SCDMA} DMA is started on the radio frame boundary of each AxC where each OFDM symbol is separated into a unique packet. For {GSM, Edge2}, alignment is initially made at the 60 ms boundary, but then each timeslot is separated into a unique packet.

Figure 4-4. WCDMA Timing



AIF2 handles WCDMA timestamps similar to other standards, but the DMA is different. Neither AIF2 nor RAC has sufficient buffering to align input data to the radio frame boundary. As a result, AIF2 performs a continuous DMA of WCDMA data to RAC with only alignment programmed to the granularity of 32 or 64 chips. The DMA is not aware of the radio frame boundary and can start with somewhat random alignment to radio framing.

TAC always begins transmission on radio frame boundaries. The DMA is not aware of radio framing, where DMA may start reading TAC output even before the beginning of the radio frame.

4.1.1 Synchronization and Resynchronization

Both timers in AIF2 (phy, rad) are synchronized to an external source and serve as the synchronization of AIF2. AIF2 startup is synchronized to these central references. If these timers are realigned during AIF2 processing, AIF2 will have some period of time of non-normal functionality, but the AIF2 will eventually recover (not more than two 10ms frames of time).

If the RadT timer gets modified on-the-fly, the AIF2 phy will run normally and only the protocol layer and DMA layer of AIF2 will be effected. Within 20 ms, AIF2 will return to normal operation after the timing reference changes.

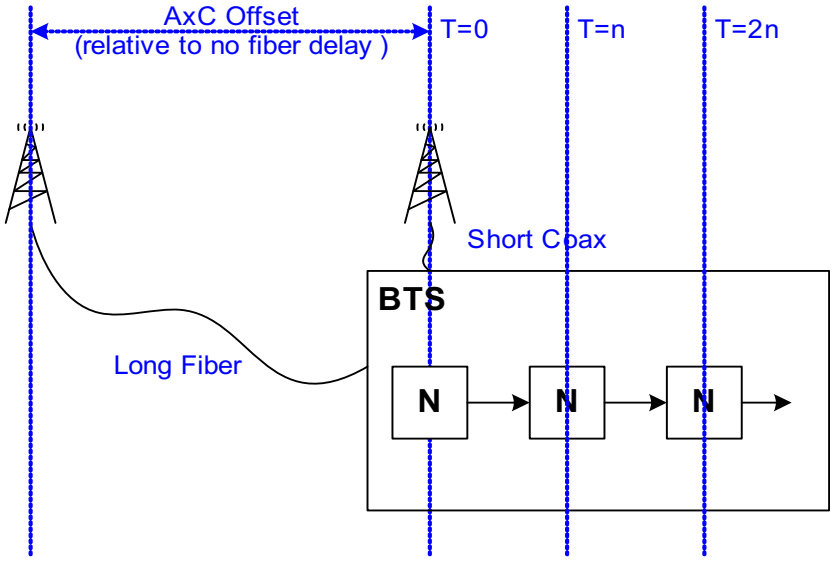
4.2 AxC Offset

If the antenna is co-located with the base station, there is no antenna carrier offset. The coaxial cable delay is treated as an extension of the air-propagation delay where the sampling at the RF card is considered to be time=0.

The first device in the OBSAI daisy chain is considered to receive the antenna data with a time=0. Each subsequent “hop” in the daisy chain has a time offset relative to the delay through the previous device.

The concept of an antenna carrier offset is new to AIF2 (not previously supported in AIF1). For legacy support of BTS with no Fiber delay, this new AxC offset parameter is programmed to be 0.

Figure 4-5. BTS AxC Offset



For a centralized BTS that is supporting remote radio heads, there is a long fiber delay between the radio head and BTS. The time it takes for the signal to propagate through the fiber is the antenna carrier offset and is relative to a perfect connection (as if there were no delay at all).

The programming of AxC offset is a fixed value. Each hop of the daisy chain, the AxC offset is programmed with a different value to compensate for the time propagation through each daisy chain node.

If Ingress AxC offset is zero, PD will start processing when it detects the first AxC data within the AxC offset window boundary. If PD fails to find the first right AxC data within the window, the PD link will not work until it meets the next radio frame boundary. On the Egress side, the minimum AxC offset can not be zero because the PE channel should be turned on after the channel data transfer is ready; so the min AxC offset will be the PE2 event offset and additional fiber delay (4chip, 8chip, ...) is added based on that value.

The AxC offset usage could be different depending on user's protocol (OBSAI, CPRI) and radio standard (WCDMA, OFDM). For OBSAI, Both Egress and Ingress AxC offset unit is byte clock and AxC offset window is used to check receiving frame boundary and if it has TS = 0 value OBSAI slot inside the window range. Ingress AxC offset is the center of the window and user can flexibly expand the range of the window.

For CPRI, Egress AxC offset unit is byte clock but Ingress AxC offset unit is 4 CPRI samples (same to QW per AxC) and Ingress AxC offset is not used for WCDMA (DIO).instead, DIO_OFFSET and RAC will control the external data delay. AxC offset window concept is not used for CPRI.

.

4.3 OBSAI Reception Timing

OBSAI specifies three characteristics to define Rx timing:

- A timestamp mechanism in the PHY (RP3) data path
- Radio timer in the AT (RP1)
- AxC offset programmed into the PHY (RP3) receiver

OBSAI specifies that the timestamp is zero at the beginning of each AxC radio frame boundary. In a perfect world, this timestamp mechanism could be used for fully identifying radio frame boundaries in the OBSAI message stream, but there are multiple problems:

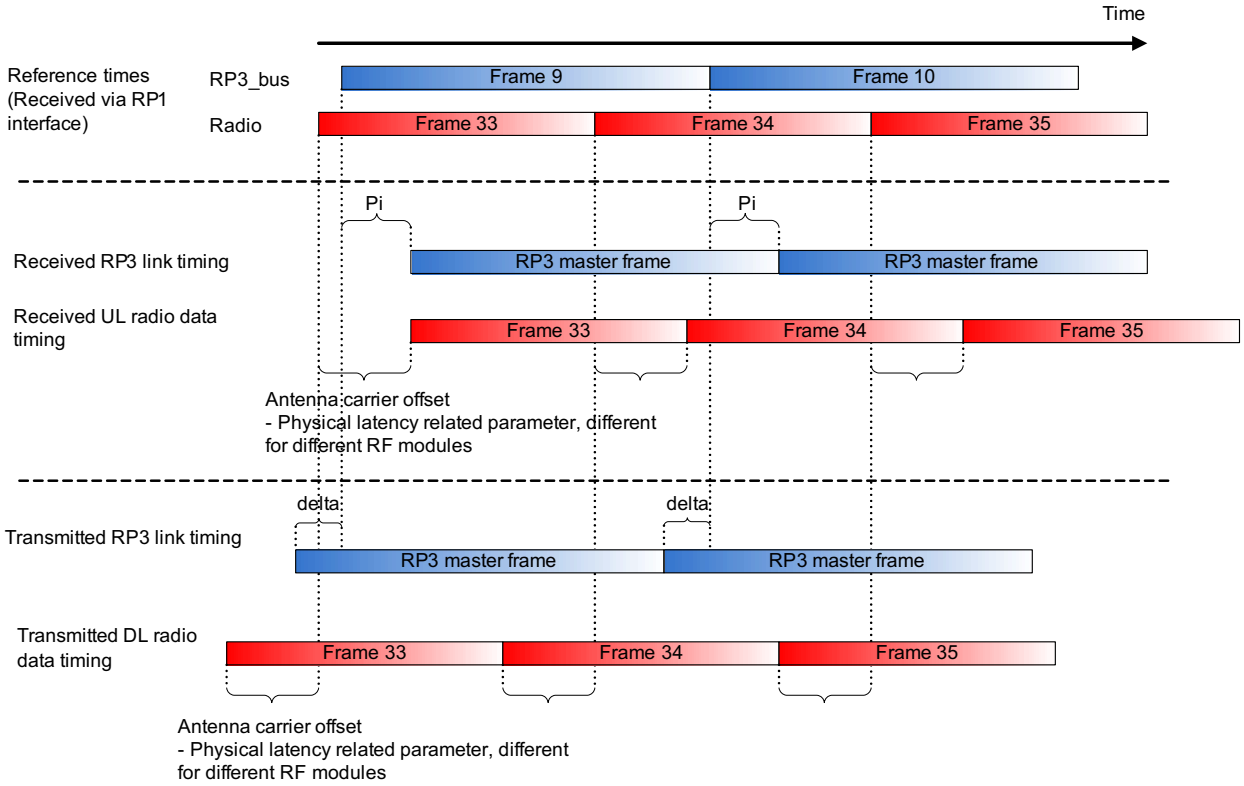
- The Timestamp field is only six bits, counting 0-to-63 and wraps after the terminal count. For radio standards with an even number of TS wraps, a TS=0 can either be a frame boundary or a TS wrap.
- SerDes bit errors can “kill” an OBSAI message. If the TS=0 frame boundary were “killed”, it could confuse the training mechanism.

Due to legacy issues, it was not possible to have OBSAI redefine the timestamp mechanism for other standards.

4.3.1 Pi and Delta Timing

Figure 4-6 shows the basic time relation of Pi and Delta for Rad, Ul, and Dl. For UL, AxC offset includes Pi offset. The AxC offset shows the time difference between real UL radio data start time and the reference radio frame boundary. For DL, the AxC offset shows the time difference between real DL radio data start time and the reference radio frame boundary

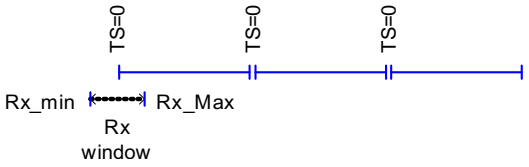
Figure 4-6. Pi and Delta Timing Example



4.3.2 OBSAI Radio Timing

Antenna carriers, transported over the OBSAI Protocol, have embedded radio timing boundaries. For {WCDMA, LTE, WiMax, TD-SCDMA} radio standards, the OBSAI timestamp equals zero on the radio frame boundary. The timestamp also equals zero every time the value wraps. The challenge with OBSAI reception timing is to determine which of the TS=0 represents the radio frame boundary.

Figure 4-7. OBSAI Radio Timing

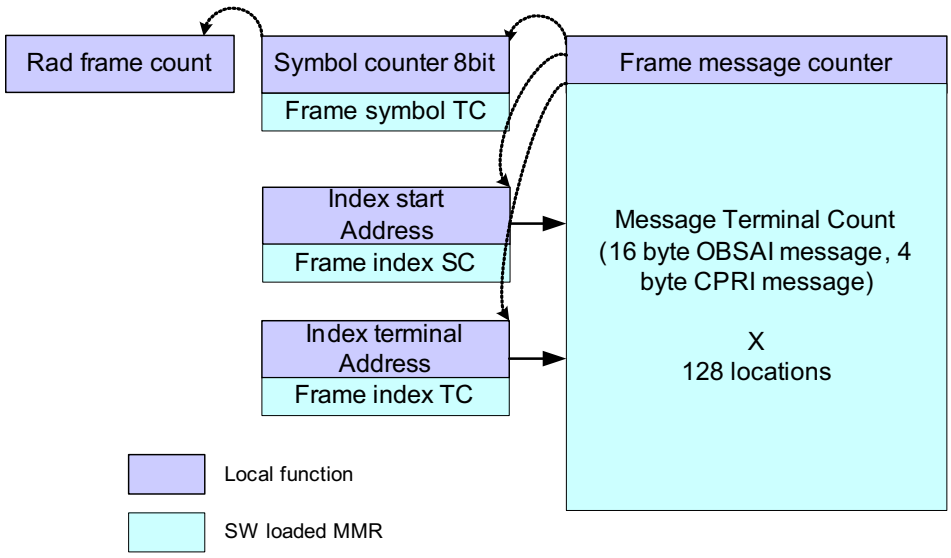


The application software has previous knowledge of approximate expected timing. This expected timing is referred to as the *antenna carrier offset*. Figure 4-7 shows that OBSAI radio timing could be captured by the Rx AxC offset window and this is the same mechanism as the Pi window for the physical timing. The user can set the AxC offset window size for ingress and AxC offset is the center of the window.

4.3.3 Tx Timing Message Construction: Radio Counter

The RadT timers from the previous section are intentionally simplified to facilitate offset for window creation. The counters do not yield a symbol number or timeslot. The AxC frame boundary (which is made by the AxC offset) is used to start a RadT timer per AxC (which does break down the count value into slots/symbols); these Radio framing counters are incremented once per received message for each symbol or slot (WCDMA).

Figure 4-8. Tx Framing Counter Circuit



The frame message counter counts messages (one for every CPRI sample or 16 bytes per OBSAI message). A separate counter is maintained per symbol; that's why we have 128 locations for frame message TC lut. Frame index SC, TC is used for the special purpose of supporting six different LTE versions; it divides the whole frame message TC lut region into six separate regions.

The terminal count of the frame message counter is used for breaking the data stream into symbol packets. The Symbol/Slot count (DMA packet index) is appended to the header (descriptor) of these packets and is used by software to check that the DMA packet is aligned to the radio timing. AIF2 fails the AxC if the DMA packet index does not match the expected symbol number (not applied to WCDMA).

AIF2 simultaneously supports extended and normal cyclic prefix LTE symbol sizes and supports three different LTE bandwidths. Effectively, PD and PE support six different sets of terminal counts to support the LTE variations. There are six different Symbol TC registers to support this. Because LTE only uses a maximum of seven terminal counts per configuration, the frame msg TC lut is not duplicated, but rather, different regions of it are allocated (by the user). The user configures this via six different sets of “start count” and “terminal count” parameters in the frame count register for the lut index.

4.3.4 AIF2 Implementation of OBSAI Radio Timing

The two timers {PhyT, RadT} in the AT module are synchronized externally to a “golden” timing source. These timers are the timing reference for all Rx and Tx timing operations.

4.3.4.1 AIF2 OBSAI Ingress Reception Timing Implementation

4.3.4.1.1 OBSAI RX Timing: Pi Handling

The RM detects the presence of the K-characters and the result of this comparison is transferred (with optimal latency) to the AT. Later, the AT determines if the last K-character was a frame boundary demarcation and sends this information to the RM. The reason for the two signals is in regards to timing accuracy. The first signal is extremely accurate while the second potentially has a degree of timing error or latency associated with it.

The AT uses the strobes from RM to determine if the frame boundary arrived at the expected time, and to capture the PhyT timer value at the time of frame boundary. This check and capture are then further qualified with the indication of second strobe indicating the frame boundary.

4.3.4.1.2 OBSAI Rx Timing: Radio Timing Alignment

OBSAI timestamp information gives framing information, where the timestamp is zero on a radio frame boundary . The timestamp is a six-bit field that starts at zero and increments throughout a radio frame; during this count, the value wraps back to zero multiple times.

The main objective in finding radio timing alignment is to create an approximate window or range where TS=0 is expected; the next step is to consider TS=0 within this window to be the AxC radio frame boundary. This operation is completed only once to synchronize the frame FSM (for each AxC) in the PD; subsequent framing is maintained by the Frame FSM.

Application software gives AIF2 the AxC Offset for each antenna carrier. The AxC Offset is used to create the compare window.

4.3.4.1.3 OBSAI Rx Timing: Symbol/Slot Boundary Identification

The AIF2 Receive circuit will maintain a virtual counter per AxC, counting through the radio standard frame protocol to identify LTE or WiMax Symbols boundaries. The Radio frame boundary from previous steps identifies the beginning of the count sequence and realigns the count sequence on every radio frame boundary (on an AxC-by-AxC basis).

AIF2 supports both extended and normal LTE cyclic prefix symbol length simultaneously and supports three different LTE bandwidths simultaneously. This is supported via multiple sets of terminal counts.

The receive circuit will also maintain a TS counter per AxC that mirrors the timestamps received from the PHY. AIF2 can support two sequential messages “killed” due to SerDes bit errors. When +1 or +2 TS is detected out of order, both the TS counting circuit and the Radio Frame Protocol counter will increment by +2 or +3 positions.

- +1: Normal increment
- +2: Normal increment plus skip one message (due to SerDes bit error)
- +3: Normal increment plus skip two message (due to 2 SerDes bit error)

4.3.4.2 AIF2 OBSAI Egress Transmission Timing Implementation

For egress (transmission) timing, both message construction and radio frame timing are controlled by the AT timers.

The PhyT is used to control the OBSAI message FSM that counts through each message and K-char. This state machine changes both the modulo and Dual Bit Map FSMs. The output of the transmission rule FSMs yield the AxC index indicating which AxC is to populate the given message. The AxC number is used to index AxC specific information and specifically the AxC Offset.

Once the AxC offset is known, the AxC Offset is compared to the RadT. When the RadT has transitioned to or past the AxC offset, the radio frame boundary of the particular AxC is indicated.

The AxC Radio Frame Boundary (60 ms timeslot boundary for GSM) is used only to start a radio frame counting FSM that counts through AxC messages and OBSAI timestamps.

Framing counters are used both for error handling and in symbol boundary prediction (for non-error cases).

Sources of Timing Error:

- Dual Byte Clock
  - Random Alignment to RP1
  - Double clock, where as OBSAI message is 19 bytes (odd number of byte clocks
- TDM of AxC
- OBSAI Control Messages (throw off Radio alignment to RadT)

Due to timing errors, the exact position of the radio frame start within the OBSAI link can be off by one sample; so there are two possible starting positions of the Radio framing. This positional “error” is identical for all devices in the chain, so there is no system error with this positional difference.

4.3.4.2.1 Dual Byte Clock Error

There are two cases of possible error due to Dual-Byte Clock PLL alignment:

- No error: If the RadT is synchronized via software relative to PhyT
- One clock error: if RadT is synchronized via external pin or RP1

There is a basic assumption that the dual byte clock has random phase alignment relative to any external signals. (This is may not exactly be true depending on physical clock routing and SerDes PLL, but is the most conservative assumption.) The result of this assumption is that when RadT or PhyT are started from an external source, there is one clock cycle of uncertainty between the two timers.

The PhyT in two different devices can have different timing alignment by a factor of one dual byte clock. There is no net result from this PhyT vs. PhyT error; the received Pi is immediately normalized relative to the PhyT, so any error of the PhyT alignment has no relative effect.

If PhyT and RadT external starting strobes have random phase alignment to the PLL that is sampling these strobes, there is a possible timing difference of one clock cycle between PhyT and RadT.

An OBSAI message is at least eight dual byte clocks in duration. If the programmed AxC offset were effectively centered on the eight clock cycle window, then some small number of clocks of uncertainty can be tolerated.

An alternative approach to this timing error is to eliminate the RadT alignment altogether. One way to sync the RadT is via software writes of MMRs, which sync the RadT relative to the PhyT timer value. If this synch mechanism is used, the result is zero alignment error between RadT and PhyT (regardless of external clock quality or delay).

4.3.4.2.2 Dual Byte Clock, ½ Clock Error

Modern high-speed SerDes modules clock in two bytes at a time. The OBSAI message length is 19 bytes. With only a two-byte timing granularity an odd message length results in a 0- or one-byte clock error.

This timing error is very small and can easily be nullified by having software set the AxC Offset to be centered on the OBSAI message. That's why the Egress AxC offset min value should slightly bigger than PE2 offset time (PE link frame data generation start time).

4.3.4.2.3 TDM of AxC Error

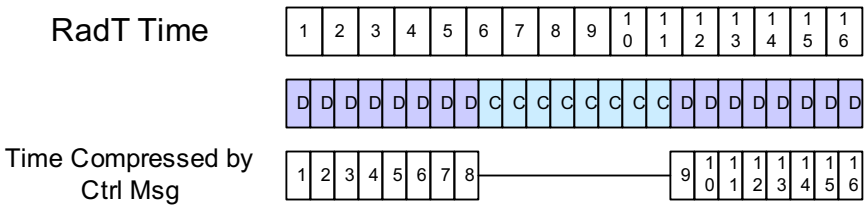
Antenna carriers are interleaved on an OBSAI link; when interleaving, there is a time difference between when the first AxC is transmitted and the last AxC is transmitted. It is very likely that the software that programs AxC offsets cannot handle the complexity of adjusting AxC offsets to compensate for OBSAI transmission order, so this TDM timing differences can be thought of as a timing error.

If this error was tolerated, in WCDMA there is a possibility of a four-chip alignment uncertainty between two different AxCs on the OBSAI link. Timestamping and framing would be correct for each AxC, but relative to the other AxC there could be a strange timing difference. (This should not be a problem because the receiver uses header-based processing).

By using dual bitmap rules instead of multiple modulo rules, the timing alignment difference between multiple AxCs is avoided. AIF2 samples and holds the RadT timer value for the set of AxC supported by a single dual bitmap rule. With multiple AxC using the same RadT timer value, all AxCs are transmitted synchronous to one another.

4.3.4.2.4 OBSAI Control Message Placement Error

Figure 4-9. How Control Messages Compress Time



OBSAI control messages are intended for passing non-antenna traffic and represent "extra" bandwidth in excess of normal antenna traffic. Because these control messages do not contain antenna traffic but are interleaved with antenna traffic, they represent an error for timing alignment.

Regardless of the link rate, the control message burst is 76.5 dual-byte clocks in duration. It would be exceedingly difficult for software to predict these control message burst locations when calculating AxC offset, so these control messages result in a timing error for message construction.

A worst-case analysis would be the 8x link rate with one modulo transmission rule. The 8x link rate has a burst of eight control messages that effectively yield a positional uncertainty of messages.

OBSAI specifies that the receiver should implement header-based processing, so the positional uncertainty has no negative impact.

4.3.4.2.5 Tx Timing: Message Construction

The timing of message construction is tightly coupled to AT PhyT. Effectively, PE has an OBSAI/CPRI FSM that counts though the two byte phases of these radio standards. The FSM is started at a fixed (programmable via PE2 event offset) time prior to Delta transmission time of the TM. The OBSAI/CPRI FSM exactly precedes the TM consumption rate by this fixed time. When FSM indicates the beginning of a message, the message is constructed.

At the beginning of each OBSAI Message, or CPRI basic frame (indicated by the OBSAI/CPRI FSM), the AT RadT is used as a time reference. RadT is used for calculation of the radio frame boundary.

4.3.4.2.6 Tx Timing: Delta Detail

Delta time is based on the PHYT. DELTA is set in TX byte clocks. Each link will have a delta event that can be adjusted only in a positive offset direction from the frame boundary.

For positive DELTA, the event offset should be set to *delta*.

For negative DELTA, the event offset should be set to *#Frame clocks – delta*. For negative DELTA, there the negative delta control bit must be set to *1*.

The frame number (BFN) will be passed to the TM submodule and is the captured PHYT Frame value when delta is reached. For a negative DELTA, the value will be PHYT Frame –1.

4.4 PhyT and RadT Timers

There are two timers in the AT (AIF2 Timer) that are time references for reception:

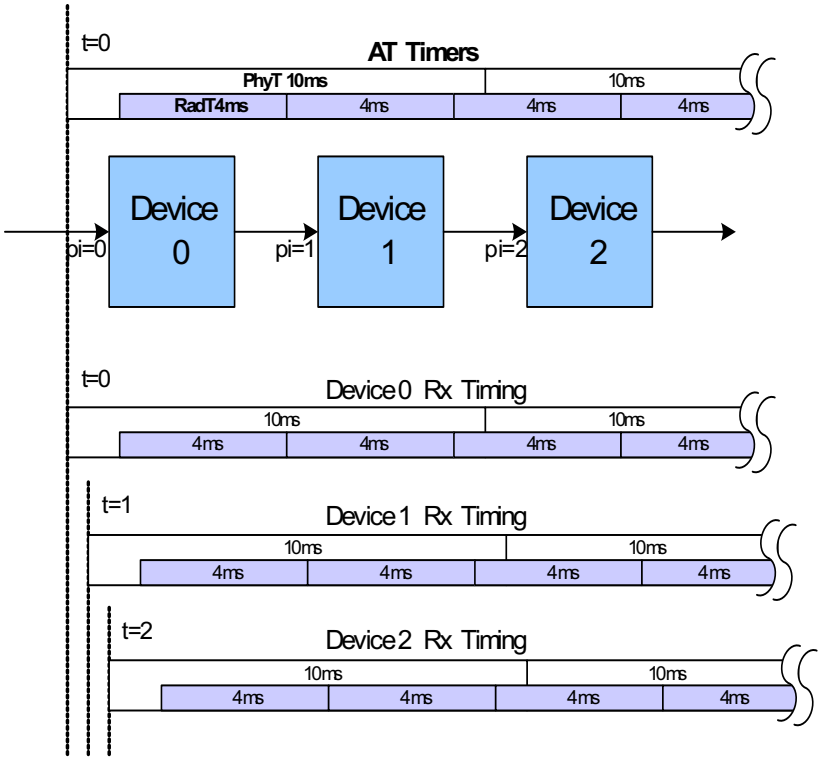
- PhyT: Physical link timer, 10ms periodic
- RadT: Radio timer, periodic with radio frame timing

These timers are synchronized to the external base station timing unit. The value of the timers is an ideal time alignment (ideal in the sense that there is no time offset, and the timers are exactly aligned to base-station time).

In an ideal sense, if a link is received at a device with Pi=0, then:

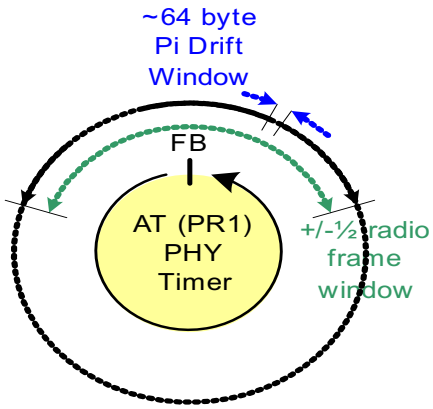
- The PhyT would be exactly aligned to the Link timing where the K28.7 would exactly align with the timer wrap.
- The RadT would be exactly aligned to the radio information embedded in the link where the RadT frame wrap would perfectly align with the frame boundary of the incoming radio traffic.

Figure 4-10. Rx UL Timing, Daisy Chain Example



With a chain of devices, only one can have a perfect Pi=0 alignment. Each hop within the daisy chain will likely have around one chip of Pi latency. These subsequent links and radio streams would be misaligned to the AT timers as a function of the observed Pi.

Figure 4-11. Valid Pi Window and Drift



Within a link, the K28.7 marks the 10 ms frame boundary. The link frame is permitted to arrive ± 5 ms relative to the AT (RP1) PHY timer frame boundary. The Link timing is permitted to drift to some limited degree. A total of 64 bytes of drift can be tolerated by AIF2.

The RM (receiver) has a 64-byte buffer that can be used to accommodate timing drift. The user effectively selects the buffering by programming the RM configuration fifo threshold field (two bits) to get proper start time for reading from the buffer after {1,4,8,16 dual bytes} is received. The allowable drift is the total 64 bytes centered on the designated, initial watermark.

4.5 AIF2 Timing Details

4.5.1 DMA Timing

In a qualitative sense, RAC and TAC are local resources with dedicated access to AIF2. L2 or shared memory is accessed by many peripherals with greater chance of contention. Users need to dedicate more slack timing for shared endpoints and less slack for dedicated endpoints.

For Egress, it is up to the user to insure that DMA data is available prior to beginning PE message construction (PE2 event). Breaking real time on DMA has similar effects to breaking CorePac MIPS real time. Specifically, AIF2 will fail each affected AxC until the beginning of the next radio frame boundary.

For Ingress, it is important that DMA is efficient enough that the AIF2 input buffers never overflow.

For DIO (WCDMA), the Ingress DB buffer size is 8 QW and can have up to a 32-chip time delay before it wraps.

The Multicore Navigator mechanism {LTE, GSM, WiMax, TD-SCDMA} manages input and output buffering more automatically, while the DIO DMA {WCDMA} is a much more manual and deterministic flow.

4.5.1.1 PKTDMA Timing

The AIF2 is a real time system, meaning that there is a limited window of opportunity for transmission. Missing this window of opportunity is known as *Breaking Real Time*. In AIF2, it is specifically *Breaking DMA Read Time*. The penalty for breaking real time is fairly severe; the affected antennas stop transmission until the beginning of the next radio frame (in many case 10 ms). During this time, it is likely that all calls on the affected antennas will be dropped. As such, Egress has much more interesting timing problem than Ingress.

QMSS (Queue Manager Sub-System) is the central resource for queuing free packet storage resources (memory management) and queuing packets that are ready for transport or consumption by endpoints. As a central resource, the QMSS blocks access to one resource while servicing other resources. The length of time for which QMSS blocks access adds to the overall transport time of a packet.

Typically engineers look at DMA bandwidth (and QMSS bandwidth) on an average basis because average bandwidth is the best understood criteria in our radio system. However, peak conditions are the usual source for breaking real time and peak conditions are difficult to characterize.

PUSH/POP: When a peripheral or CorePac wants to create a packet, it needs a memory location to start writing the data. Multicore Navigator manages memory by having a queue of free memory location (buffers) to write to. The peripheral needs to POP a location from QMSS to receive this address. With the current QMSS, approximately 9 clock cycles are required for every POP operation.

After the CorePac or peripheral creates the packet, it passes the packet to either a DMA engine or software thread for processing. The Pass operation is performed by posting a pointer to this packet on a queue. This operation is known as a PUSH and takes 12 clock cycles when 32 descriptors are used.

Table 4-1. QMSS Back-to-Back Performance

Operation	VBUS Cycles
PUSH	12
POP	45

Congestion: There are many parallel peripherals and CorePac cores that can all be performing Multicore Navigator packet operations. Congestion occurs either when there are either multiple simultaneous requestors or with a single requestor rapid firing or consuming packets. The user must handle these congestion issues in one of two ways:

- Add slack or a timing margin
- Carefully calculate and predict Multicore Navigator QMSS congestion to avoid failing cases

All modules (even a well behaved module such as FFTc) will perform a single PUSH and POP in close proximity to each other when processing back-to-back packets.

Poorly behaved peripherals such as AIF2 or SRIO will have peak packet flooding conditions where packets are machine gun fired into the system. There are two different conditions that can cause this type of behavior:

- Small packets: 16-byte packets are a worst case offender, requiring a PUSH and a POP for relatively low interface or VBUS bandwidth. Both SRIO and AIF2 are built to handle small packets (or large) as small as 16 bytes.
- Many parallel channels: AIF2 is a worst case offender for parallelism. AIF2 has up to 128 parallel channels. If these channels are all time-aligned, there is an amazing congestion of QMSS accesses.

While hardware peripherals tend to have a Multicore Navigator characteristic, software use of Multicore Navigator can be of either extreme (or more likely a mixture). Sixteen-byte packets that are received by either SRIO or AIF2 are likely delivered to the CorePac. The CorePac software needs to access QMSS in much the same way as the hardware peripherals. Even for the otherwise well-behaved uses of CorePac (such as generating FFTc traffic) there is a danger of being organized into peak QMSS accesses. To save MIPS, a software user may like to cache Multicore Navigator descriptors in an effort to expedite packet creation.

The first-order worry of QMSS congestion is breaking AIF2 real time on the transmit side (Egress side) and lesser a worry of overrunning the AIF2 receive buffers. Once Multicore Navigator packets have been started, the AIF2 implements a fair share scheduling algorithm to even out traffic among channels.

4.5.1.2 AD DIO DMA Timing

Internal system events trigger the DIO (up to three engines) to transfer data. The user calculates how much time gets allocated for the DMA transfer after the internal system event triggers.

To start, the user calculates the ideal DMA transfer time (assuming no contention). In the following tables, look up the number of VBUS cycles and multiply by the number of VBUS bursts. (Disregard the round trip time as this will simply degrade this ideal value).

Table 4-2. Egress (Tx) DIO Performance

Burst Length (QW)	Bubbles (waste)	VBUS Cycles
4	1	5
2	2	4
1	3	4

Table 4-3. Ingress (Rx) DIO Performance

Burst Length (QW)	Bubbles (waste)	VBUS Cycles
4	2	6
2	3	5
1	4	5

Then take the ideal value and apply a derating factor. For local RAC or TAC access, +50 percent derating factor is suggested. For L2 or MSMC, +200 percent is recommended. DDR access is so heavily dependent on loading that a simple rule of thumb is not supplied.

Example:

Assuming a 1 GHz CPU clock, the resulting VBUS clock is 3.3ns, yielding 157 vbus_clk allocated for the transfer.

- 32 UL AxC (8x), 8 chip iteration, two QW per AxC, 4-burst, RAC destination
  - 16 bursts of 4 = 16 x 6 = 96 VBUS clock
  - Derate by 50% = 144 vbus_clk
  - 475.2 ns + 518.1 ns(157 vbus_clk) = 306 sys_clk (307.2 MHz)
- 32 DL AxC (8x), 4 chip iteration, 1 QW per AxC, 4-Burst, TAC source
  - 8 bursts of 4 = 8 x 5 = 40 VBUS clock
  - Derate by 50% = 60 vbus_clk
  - 198 ns + 518.1 ns = 221 sys_clk (307.2MHz)

4.5.2 AIF2 Internal Delay

4.5.2.1 DB =>PE Timing

CPRI and OBSAI usage of DB fetch differs. CPRI counts on prefetching data from the DB, then storing the data into the FIFO, so user doesn't need to care about delay. In OBSAI usage, the data is usually fetched and used immediately. For this OBSAI use case, the PE needs to be configured with a delay value of 28 sys_clk.

Program Delay:

OBSAI = 28, CPRI = 0

4.5.2.2 PE=>TM Timing

- OBSAI: 30 + 28 = 58 sys_clk Minimum
- CPRI: 30 + 0 = 30 sys_clk Minimum

This is the minimum distance between the PE2 event and TM Delta. The minimum spacing between these system events represents the time it takes for PE to create link data prior to the TM beginning transmission. The PE2 event is the starting point for link construction.

The PE1 event is the starting point for link retransmission, insertion, or aggregation of RT. The minimum distance between PE1 and PE2 is 10 sys_clk cycle. PE1 and PE2 event offset could be set by using at_pe_event_offset and at_pe_event2_offset register in AT

4.5.2.3 Delta and PE Event Offset Calculation

Delta and PE event offset is measured by sys_clk (307.2MHz for OBSAI and 245.76MHz for CPRI). TM differentiates negative delta by reading at_neg_delta register setup. There are several time delay factors in the egress side. They are Multicore Navigator data transfer delay or DIO DMA delay, Egress PE internal delay for PE1 and PE2 event.

Delta (DL case) = DMA time (Multicore Navigator or DIO) + PE1=>PE2 event timing (more than 5 clock) + PE =>TM timing (less than 60 clock)

- There is approximately one chip (260 ns) of minimum latency between reception and retransmission of a link.
- The distance between RM (Pi) and RT is approximately 15 sys_clk (OBSAI).
- Pi time could be the same to PE1 event time in case of aggregation or retransmission.
- The distance between PE1 and PE2 should be bigger than 5 sys_clk (even though there's no aggregation).
- The distance between PE2 and Delta is 60 sys_clk or less.
- Total of 80 sys clock cycles (redirection delay) could be added to get Delta for retransmission.

Applies to both OBSAI and CPRI.

The maximum difference between reception and transmission timing is a function of the maximum amount of buffer built into the AIF2. AIF2 implements a 1k byte buffer (FIFO) per link (in the RT) allowing for a maximum of 1k byte clocks between reception and transmission timing (This is essentially the difference between Delta & Pi). Additionally, the RM implements a 64-byte FIFO which, when used, can add latency to the latency between reception and transmission.

For AIF1, there was a need to set different clock value for different link rate (1x, 2x, 4x) but AIF2 uses only the byte clock (sys_clk) for delta, pi setup. 8x just uses this frequency as data cycle but 4x holds data for two cycles and 2x holds data for 4 cycles internally. Programming is earlier because Delta and Pi are in the same unit for the 2x, 4x, 8x link rates.

4.5.2.4 TM =>RM SerDes Loopback Timing

Using SerDes loopback mode, it is 15 clock cycles (4x link rate) between TM delta and the resulting observed pi at the RM.

4.5.2.5 TM =>RM Timing

The RM requires setting the PiMIN and PiMAX. Given that there is significant latency after TM delta, it is recommended to set PiMIN = Delta for the transmitting side. Setting PiMAX is more complex. The smallest recommended PiMAX is delta +10, 15 or 25 (8x, 4x or 2x) for a system where devices are physically close and board trace is tightly controlled. For systems with long fiber delay, the user must calculate the expected propagation delay and program PiMAX accordingly.

If real received frame boundary is within the window range, RM module will detect it automatically. Normally, If Pi is negative, the Pi min value could be bigger than Pi max (e.g. Pi min: 3071989, Pi max: 10). When RM detect this is true, it handles this as negative Pi and the Pi offset will be checked around (before) the frame boundary.

4.5.2.6 RM => RT timing (Retransmission or Aggregation)

The propagation delay from RM to RT is 15 clock cycles (OBSAI). This is the time from Pi arrival as measured at the AT until data is available to aggregate with PE.

The PE1 system event is a preparation signal for RT. When aggregation is used, the PE1 event should precede both the PiMIN and PE2 event. Set PE1 equal to PiMIN where PiMIN represents the latest time at which the frame boundary is expected. (Because of pipeline delays, there is no need to add a gap between PiMIN and PE1).

NOTE: While timing of the PE1 strobe is rather loose, there is a maximum value for the system event. RT can not accept a PE1 event strobe more than 20 message groups ahead of a frame boundary. This is an extreme value with no practical value. Reasonable upper limits of PE1 (relative to Delta) are {500clk, 500clk, 1000clk, 2000clk} for respective link rates of {8x, 5x, 4x, 2x}. This practical upper limit is dictated by the maximum supported offset between Pi & Delta.

4.5.2.7 RM =>PD =>DB Timing (UL Case)

The propagation delay from RM to PD is four clock cycles.

The PD pipeline itself is eight clocks long plus a possible five clocks for having six channels share a single port into DB. The largest component of ingress timing has to do with packing the PHY standard {OBSAI, CPRI} and RSA data format.

In CPRI (non-RSA), a full quadword is accumulated before writing to DB. Because AxCs are interleaved on a word-by-word basis, four samples of time must elapse before DB receives the first data. Clearly four samples of time is radio-standard dependent. With OBSAI, the AxCs are interleaved qwd-by-qwd. Again, within four samples of time, all AxCs will capture a qwd.

With RSA data format, PD will wait until two quad words of data for a given AxC are received before writing to DB, then PD will write out both quadwords sequentially.

The largest impact of timing is the fact that all six links of PD share a single write port to DB. The physical DB memory is in the vbus_clk domain so writes to DB occur in the slower of the two clock domains {sys_clk, vbus_clk}. The following factors contribute to this issue:

- Many high speed links
- Many AxC
- RSA data format (RAC)
- CPRI
- Slow vbus_clk

In practice, setting 20 clocks for all these kinds of internal delay and fuzzy factors avoids complex calculation. So total Min Ingress Delay before starting First Ingress DIO DMA event will be **Pi + 20 + 160 clock time margin for PD,DB timing (80 clock for DL data type) + Maximum Ingress 4chip or 8chip accumulating DMA time**).

NOTE: In DIO mode of operation, DMA is triggered by precise timed internal system events and therefore has to be correctly timing characterized. In Multicore Navigator modes of operation, the DMA is triggered by data arrival making these timing calculations mostly unnecessary.

4.5.2.8 MAC Timing (SerDes Timing)

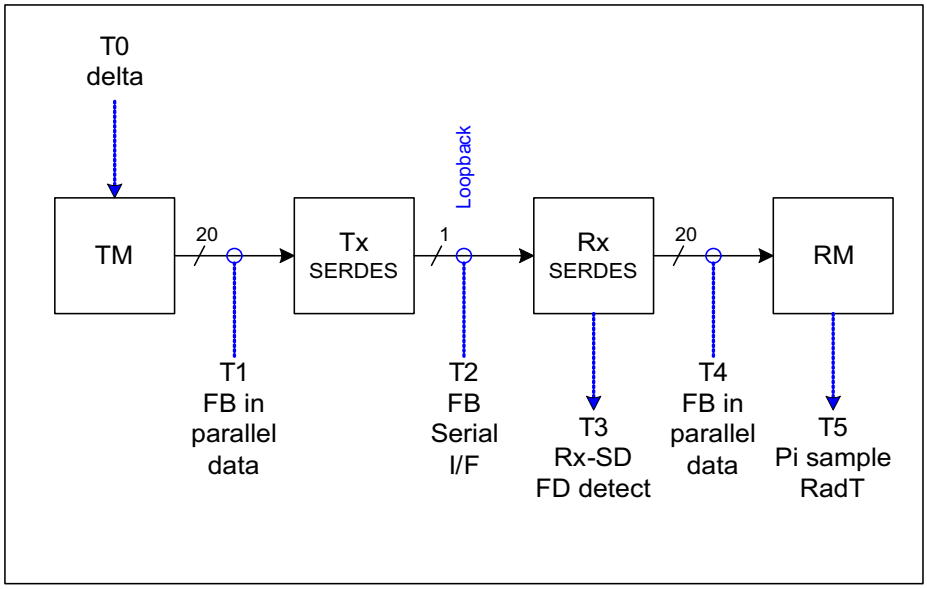
MAC level timing measurements are made with OBSAI PHY standards for {2x, 4x, 8x} link rates. Timing measurements are made in SYS_CLK. SYS_CLK are highly appropriate as they are an exact multiple of the sample clock. There are {1, 2, 2, 4} sys_clk corresponding per byte for the following link rates {8x, 5x, 4x, 2x}. SYS_CLK runs at 307.2HMz for OBSAI and 5x CPRI; SYS_CLK runs at 245.7MHz for CPRI {2x, 4x, 8x}.

Timing measurements are made using SerDes loopback. All measurements are relative to TM delta.

Sources of error in the timing measurements are:

- Tx-Rx Rx-Tx clock domain crossing
- Potential clock misalignment due to two-byte interface

Figure 4-12. AIF2 MAC Timing Measurements



Measurements are made at four different points (relative to TM Delta):

- T1: observation of K-char in two-byte data stream feeding TX-SerDes
- T2: RX-SerDes reporting of PHY frame boundary
- T3: observation of K-char in two-byte data from RX-SerDes
- T4: Sampling of RADT timer by RM corresponding to Frame Boundary

Table 4-4. AIF2 MAC Timing Measurements

Timing Measurement	2x Link rate (sys_clk)	4x Link rate (sys_clk)	8x Link rate (sys_clk)
TM_Delta (T0)	0	0	0
FB in TX two-byte data (T1)	2	2	1
FB at Serial Interface (TX & RX)(T2)	5	3.5	3.5
RX SerDes FB Detect (T3)	11	8	5
FB in RX two-byte data (T4)	7	6	5
FB sampling of RadT, Pi measurement (T5)	24	15	10

Each row shows accumulated number of sys_clk value for each step. T3 values looks bigger than T4 values because the only signal we get from the SerDes that indicates T3 is satisfied with a status signal, and it turns out the SerDes drives the data path quicker than it does the status signal. The values in Table 4-4 for T3 show the state when the signal goes active. T5 shows the final time distance between Delta and Pi during loopback. Users can calculate the size of Pi window based on this information.

4.5.3 WCDMA Timing Example

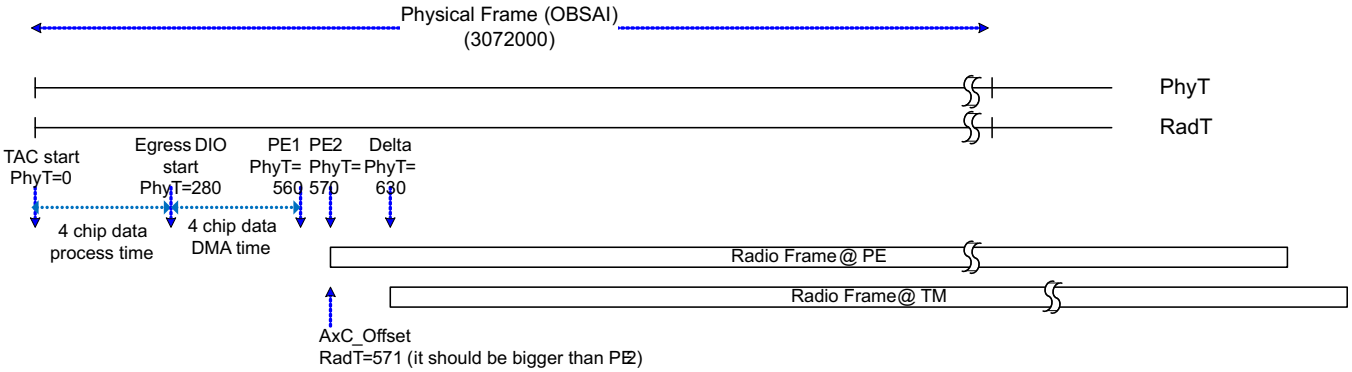
The first example is a WCDMA using DIO for Egress TAC emulation. The TAC process starts at PhyT=0 and RadT =0 and at PhyT = 280, Egress DIO DMA starts, where approximately four chips of time are given to allow the DMA to complete (which is an excessive large amount of time for the transfer to complete).

At PhyT=570, both the link and the AxC are activated. The AxC_Offset is chosen using the exact method, a value of 570 which is the same to PE2 offset. The exact method is chosen for simulation purposes such that the AxC starts up exactly when the link turns on. Calculating exact timing can be quite tricky in OBSAI as the calculation is complicated by the position of control slots. Using the approximate method is recommended for field use where it is not critical to start real traffic in the first frame of boot.

The PE1 system event in this example proved non-critical as there is no aggregation in this example (it is important the PE1 precedes PE2 by at least five clk). For systems with aggregation (RT) it is important to set PE1 to precede the PiMin.

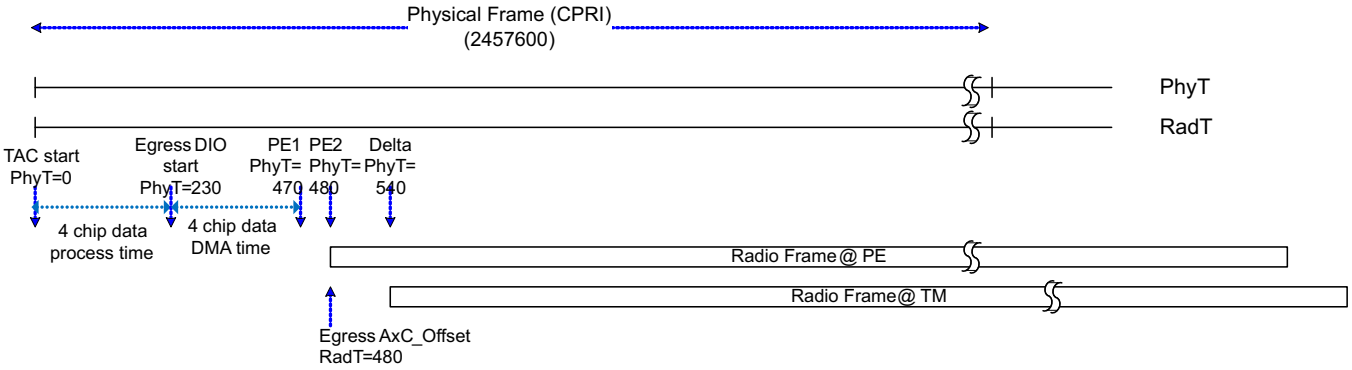
The distance between PE2 and Delta has been chosen as the minimum for OBSAI (60). It is recommended to use the Min values always (no reason to vary this timing).

Figure 4-13. WCDMA Egress Timing (OBSAI)



In case of CPRI, all other factors are the same except the byte clock rate (2457600 byte clock per frame) One WCDMA chip length is 64 clocks instead of 80 clocks (OBSAI).

Figure 4-14. WCDMA Egress Timing (CPRI)



Second example is a WCDMA using DIO for Ingress RAC emulation, the PiMin is set to 630. The PiMax is set to PiMin + 20, giving some slack in reception timing.

With reception, (after PiMax) we wait for four chips of time for two qwd per AxC (UL RSA) to accumulate then add ingress timing margin of 20 + 160 clocks. The 20 clock cycles represents pipeline delays and TDM of the DB bus (assuming we're provisioning for all links on). The 160 clocks of slack is a very broad swag number for RSA data type.

Data is transferred between PD and DB as two quad words per AxC. Because OBSAI messages require a min of 9*2 sys_clk per message (8x link rate) and there is a worst case of six links, there is a worst case of 6 DB qwd written every 9*2 sys_clk. DB is written in the slower of vbus_clk or sys_clk, but the vbus_clk has a min value of 233MHz. This translates that the 160 clks of timing margin are not required for this example.

RAC strobe (event 9 or 10) can be started after 32-chip time since the Ingress DIO DMA 8-chip event activated. For CPRI, Ingress AxC offset is not used and all external Antenna delay will be controlled by RAC.

Figure 4-15. WCDMA Ingress Timing (OBSAI)

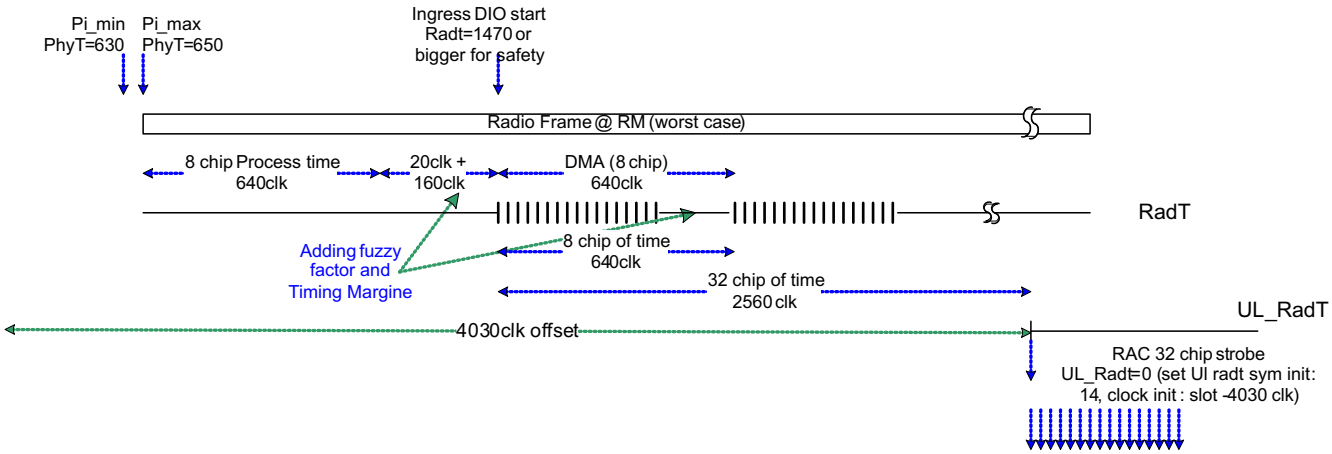
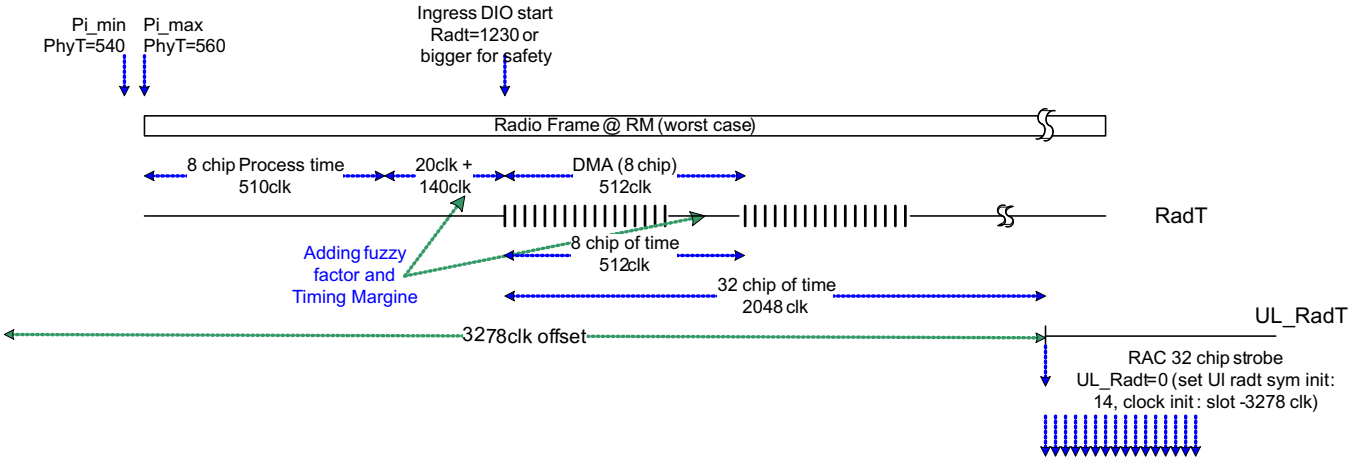


Figure 4-16. WCDMA Ingress Timing (CPRI)

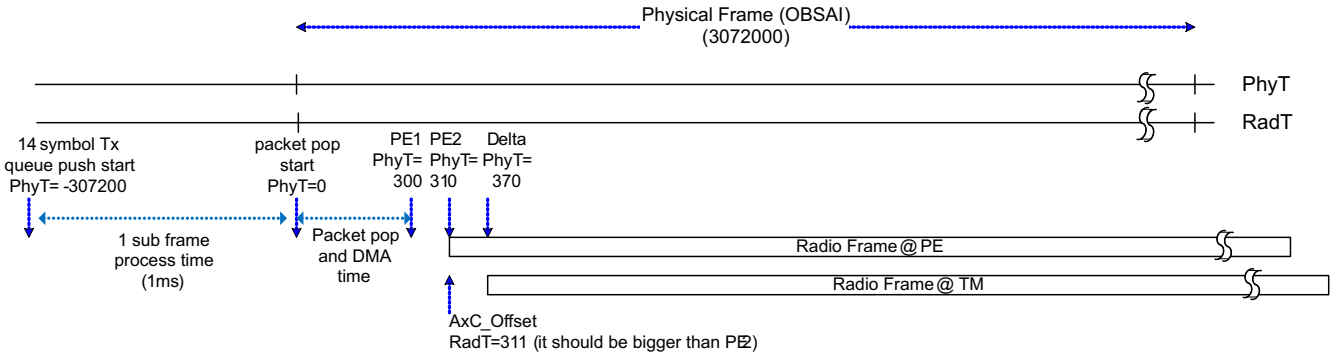


4.5.4 LTE Timing Example

LTE example is simpler than WCDMA case because Navigator will control data transfer automatically and we do not need to care about DMA event like DIO cases. The only problematic timing is 14 symbol Tx queue push time. LTE has 10 sub frames and it has 1 ms period between events. Before AIF2 PKTDMA pops packet, those 14 symbols should be pushed into the Tx queue and that's why AT support 1 ms sub frame size event for application. User doesn't need to concern about how to push 14(or 12) symbols before starting AT, because during first frame time, AIF2 phy modules (RM) is not activated, so user can get fully enough amount of time (one or two frame time) to prepare first symbol chunk push.

PKTDMA packet pop operation starts at PhyT=0 and RadT =0 and at PhyT = 310, both the link and the AxC are activated (which is a very large amount of time for the packet popping and transfer to complete). The AxC_Offset is chosen using the exact method, a value of 310 (same to PE2 event offset) is due to a pipeline delay between AD & PE. The PE1 system event in this example proved non-critical as there is no aggregation in this example.

Figure 4-17. LTE Egress Timing (OBSAI)

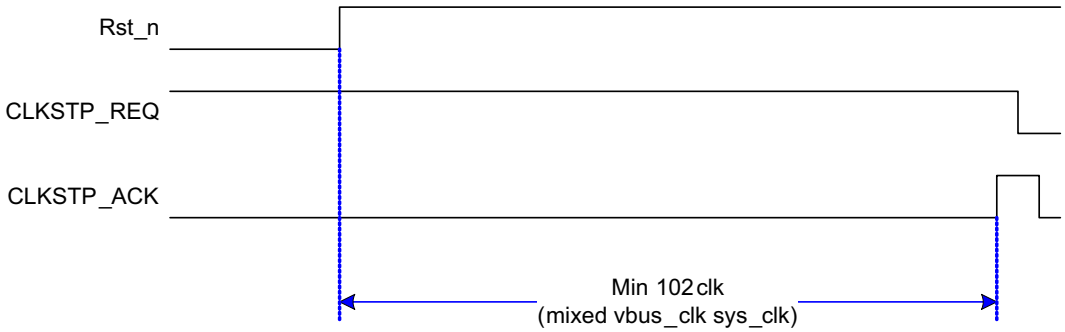


4.5.5 CLKSTP_REQ Min Timing

The CLKSTP_REQ signal (HW internal signal) is remains high after rst_n. In normal reset cases, AIF2 is tolerant of the CLKSTP_REQ activation for a short, deterministic number of clock cycles. For a true CLKSTP_REQ, software will deassert the CLKSTP_REQ signal (by setting freerun field in Aif2_Emulation_Control register) at any random time without regards to the CLKSTP_ACK signal state.

Rather than verifying every random alignment, AIF2 requires that software wait enough clock cycles for AIF2 to complete its shutdown protocol. The protocol is quick and deterministic because rst_n was just performed.

Figure 4-18. CLKSTP_REQ Min Timing



AIF2 requires the software to loop for 200 clock cycles after deassertion of rst_n. The 200 clock cycles represents 102 mixed vbus_clk/sys_clk cycles, guard banded for any reasonable clock frequency variations a system may have.

Interface Standards CPRI, OBSAI Specifics

Topic **Page**

5.1 CPRI/OBSAI Overlay of Functionality **82**
5.2 CPRI Specifics **85**
5.3 OBSAI Specifics **91**

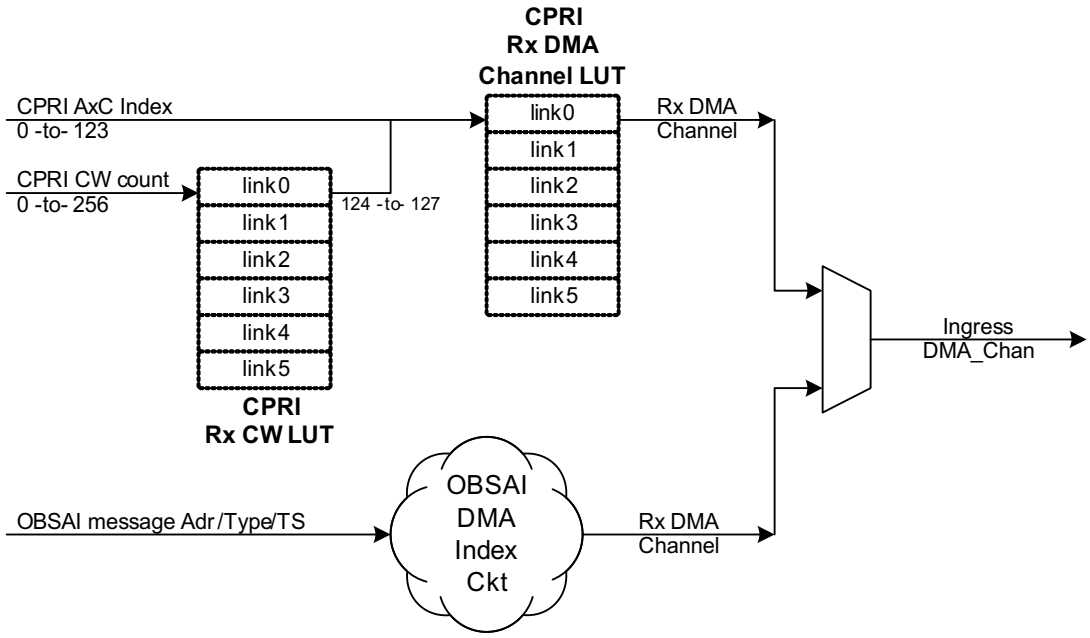
5.1 CPRI/OBSAI Overlay of Functionality

5.1.1 CPRI/OBSAI Ingress DMA Channel Addressing/Indexing

Both OBSAI and CPRI have parallel streams of data on a given link. Each standard deals with this parallelism in a very different way:

- OBSAI has a micro-packet concept where each “OBSAI message” has an addressing scheme that maps to an internal address.
- CPRI uses positional based (predefined position for different streams of data).

Figure 5-1. Ingress DMA Channel Addressing/Indexing



To overlay the two different addressing schemes into AIF2 generalized hardware, two different mechanisms are mixed depending on the interface standard being supported.

As part of the process of protocol decoding ingress links, AIF2 enumerates all possible AxC packing options and control streams with an index 0-127 per link. This link AxC index then indexes to the CPRI Ingress DMA Channel LUT which indicates:

- Whether the AxC/Control stream is active/inactive
- Which DMA channel is mapped to the CPRI Data Stream

5.1.1.1 CPRI

A CPRI link can contain AxC (Antenna Carriers) and CW (control words). AIF2 internally indexes the AxC in the order that they are extracted from the CPRI link. AIF2 supports a CW LUT, allowing the CW stream to be partitioned into {0, 1, 2, 3} substreams; each of these CW substreams is indexed {124, 125, 126, 127}, if channel {0 ~ 123} is used for AxC stream.

The CPRI DMA LUT remaps the internally indexing of streams/substreams into a DMA channel index. There are 128 different LUT possible addresses for each link (a different LUT per AIF2 link); Mapping to a total of 128 Ingress DMA channels. It is possible to map multiple streams to a single DMA channel.

It is illegal to map streams from different links to the same DMA channel. Incorrect AIF2 operation is likely to result. AIF2 can tolerate a link to contain a maximum of 124 streams. If a link contains more than 124 AxC, AIF2 has no possible visibility of those in a higher position than AxC.

5.1.1.2 OBSAI

The OBSAI mechanism is more like a CAM (Content Addressable Memory). The OBSAI adr/type/TS address space is huge and impractical to implement in a RAM LUT. Instead, 128 different compare circuits are implemented in OBSAI where the Adr/Type/TS usage is individually programmed for the 128 possible DMA channels via AIF2 MMRs.

5.1.2 Reception Timing

Prior to reception, the AT PHY Timer is started by an external time source. This timer counts through frames (using the SerDes two-byte clock).

The RM detects the presents of K28.5 or K28.7 framing characters of an incoming link. It pulses a single high (for a minimum of 260 ns) passing this signal directly from its byte clock domain to the AT. The AT double resynchronizes the signal and samples the System Timer at the high transition of the single edge.

AT checks the K-char fell within the expected Pi window specified by Pi_{min} and Pi_{max}. It alerts RM in the event that the check failed.

The RM separately determines if the K-character was on a frame boundary. The reason for separate K-char signal and frame boundary is that frame boundary identification occurs in a different portion of RM circuitry. If Frame Boundary is indicated, the shadow copy of the captured PhyT is latched into the $Pi_{captured}$ MMR. This captured value may be read by the Host as a Pi measurement.

5.1.3 Transmission Based Timing

OBSAI has the Pi and Delta concepts where reception is somewhat variable and transmission is precisely fixed. With the OBSAI approach, the expected timing at each node on a board is hand calculated and then the Pi and Delta parameters are programmed. The Pi window allows for drift and variation, but the transmission timing is fixed.

CPRI implies that transmission should occur as soon as possible after reception; CPRI has the conflicting requirement that RTT measurements should be extremely precise with an error margin of about 16 ns. In practice, a BTS uses a cascade or daisy chain of devices. At each hop in the chain, there is a clock domain crossing between Rx SerDes and Tx SerDes. With modern SerDes being two bytes wide, this represents a two-byte clock timing uncertainty at each node.

Non-WCDMA AxC data is packetized within the DB in the sense that DB marks a SOP and EOP packet when buffering in internal FIFOs. In addition to SOP and EOP, non-WCDMA radio standards are supported with a form of “count” (symbol or timeslot index) that indicates ordering of the packets relative to radio frame boundary (necessary for supporting AxC-by-AxC offsets).

The following are variations of this scheme for the different radio standards:

- **Control Packets**: only contain SOP and EOP indicators.
  - **CRC packets**: packets with a CRC are DMA’ed with a gap for the CRC location. PE will overwrite the gap with a calculated CRC.
  - **Ethernet packets**: CRC handled as above; Preamble and SOP are not represented in DMA packets and are added by PE.
- **LTE, TDSCDMA, WiMax**: Every symbol is treated as a packet by both Multicore Navigator and DB. SOP and EOP mark beginning and end of the packets in the DB FIFO. Additionally, a Symbol number is passed with the SOP. This Symbol number starts at zero on Radio Frame Boundary and increments every subsequent Symbol.
- **GSM**: Same as OFDM (LTE or WiMAx) except timeslots are treated as packets.

PE uses the symbol or timeslot index for aligning DMA data to radio timing, ensuring that data generation is correctly synchronized to radio time framing.

For WCDMA data, the concept of AxC offsets if effectively removed. For ingress, the software programs PD with AxC offset. PD uses the AxC offset to align all AxC in unison to the top of the circular buffers.

Once a control stream (DMA channel) is associated with an active PE link, there is no purging or synchronization operation performed. Control packets are transported on-demand where the transport bandwidth is dictated by transmission rules.

For {LTE, TDSCDMA, WiMax, GSM} the DMA packets have an embedded “Symbol” number in the Multicore Navigator header (descriptor). The AD (AIF2 DMA submodule) and DB preserve and pass this Symbol number to PE. At startup or for resynchronization, AIF2 and PKTDMA will perform a “quick purge” of packets until a beginning of frame is encountered (Symbol=0). Then the packets start being loaded into the DB. PE will attempt synchronization and transport of the OFDM/GSM data, checking each subsequent packet for Symbol alignment.

5.1.4 Short Frame Mode

Short Frame Mode is a feature that is used to accelerate some development tests. The PHY frame is shortened from 10ms to small fraction of 0.04 ms in order to accelerate simulations.

The length of the short frame is specifically chosen to be at least 32 WCDMA chips in length in order that the WCDMA circular buffering scheme will wrap on the short frame boundary

5.1.4.1 OBSAI

(Assuming we supported 2x rate mode) For WCDMA, a message group contains 10 messages per AxC, which translates into 40 samples. The least common multiple of these is 640 samples.

This would set the 2x short frame at 16 message groups. 4x and 8x follow the same trend thus short frame should have the following number of message groups:

Table 5-1. Short Frame Length, OBSAI Message Group

Link Rate	Length (message groups)
2x	16
4x	32
8x	64

Because the number of message groups for a full frame is 1920 * link_rate, having short frames be 8 * link_rate should not be an issue; in fact, this allows all links to have the same frame period regardless of rate.

5.1.4.2 CPRI

The hyperframe can not be shortened because the special control bytes based on the basic frame number. Each hyperframe has 256 samples regardless of rate. One hyperframe satisfies the n*32 rules, but the fact that there would not be a hyperframe boundary that isn't a master frame boundary is bad for coverage. Therefore, the CPRI short master frame consists of two Hyperframes.

For AIF2, the PHY timing and Radio Timing is very separate. Just because the PHY frames are being shortened there is no requirement or to shorten the radio frames as well. The Short Frame mode is a mechanism that only shortens the PHY timing. There are already highly programmable mechanisms in the AT that allow variable length (that is, short) radio framing.

5.2 CPRI Specifics

5.2.1 CPRI Control Data Flow

AIF2 allows control data (packet data) flow over either OBSAI or CPRI (usage of the AIF2). The characteristic of control data is its burst or on-demand data flow whereas antenna traffic is regular streaming data.

The OBSAI standard has specified a very powerful control data (packet-switched traffic) approach. AIF2 fully supports these OBSAI requirements and overlays some of the supporting hardware with CPRI usage. In particular, CPRI also allows for very high bandwidth by allowing unused AxC slots to pass control data or generic packet data.

Support of control data in CPRI antenna carrier slots and Fast Ethernet 4B/5B encoding are a particular challenge in that control data is not byte-aligned. In these modes, the AIF2 performs heavy bit manipulations for packing and unpacking data.

CPRI specifies two different mechanisms in the control word portion of the link

- Slow C&M (HDLC) ← Not supported by AIF2
- Fast C&M (Fast Ethernet ← Supported and extended by AIF2

5.2.1.1 Fast Ethernet

Fast Ethernet is Ethernet that is 4B/5B encoded. 4B/5B encoding is the encoding on top of the SerDes 8B/10B encoding. In essence, AIF2 deals with 8B/10B encoding at the PHY layer and 4B/5Bencoding at the Protocol layer.

Figure 5-2. Ethernet Frame Structure

Preamble	Start of Frame	Dst Adr	Src Adr	Length	Payload	CRC
7 bytes	1 bytes	6 bytes	6 bytes	2 bytes	1-1500 byte	4 byte

- Preamble: Constant value: 8’b1010_1010
  - Ingress: Stripped off by AIF2
  - Egress: Added by AIF2
- Start of Frame: Constant value: 8’b1010_1011
  - Ingress: Stripped off by AIF2
  - Egress: Added by AIF2
- Length Field: AIF2 does not use or check the length field of Ethernet.
  - Ingress: Ethernet frame is defined by either delimiter byte or 4B/5B encoding
  - Egress: Packet length is controlled by Multicore Navigator header
- Other Fields: Simply passed by AIF2
- Interframe gap (not shown in Figure 5-2)
  - Does not apply to CPRI, Gapping is not required over Phy
- Extension fields
  - Does not apply to CPRI, Not supported.

AIF2 performs the following operations:

- Ingress
  - Deinterleave CPRI Streams
  - 4B/5B decode
  - Parse stream into Ethernet packets
  - Strip Ethernet preamble and SOF fields
  - Encapsulation into Multicore Navigator packets
  - PKTDMA
- Egress
  - PKTDMA
  - Extract from Multicore Navigator Packet
  - Add Ethernet preamble and SOF fields
  - 4B/5B encode
  - Insert into CPRI Stream
  - Interleave CPRI Streams

5.2.1.2 4B/5B Encoding

4B/5B encoding is a part of Fast Ethernet, but AIF2 allows for the use of 4B/5B encoding simply as a packet delimiter. This special encoding can also be used as a packet delimiter for generic packet traffic that uses the AxC data slot in generic packet mode.

4B/5B encoding is a 20 percent overhead scheme where each byte is broken into two nibbles and each nibble is represented by five bits. With the extra redundancy, extra characters can be defined. AIF2 allows for the use of 4B/5B encoding only in CPRI mode and can be used for

- Fast Ethernet (4B/5B required)
- Generic CPRI control packet delimiter

The SOP is uniquely identified by a pair SSD#1 and SSD#2 (bit pattern 11000_10001) which is used by AIF2 as a comma alignment character. The EOP is identified by the pair ESD#1 and ESD#2 (bit pattern 01101_00111).

Table 5-2. Fast Ethernet 4B/5B Encoding

Name	4b	5b	Description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
I	-NONE-	11111	Idle
J	-NONE-	11000	SSD part1
K	-NONE-	10001	SSD part2
T	-NONE-	01101	ESD part1
R	-NONE-	00111	ESD part2
H	-NONE-	00100	Halt

The 4B/5B encoding occurs before stream serialization (CPRI PHY packing) and before 8B/10B encoding for SerDes transmission. (4B/5B encoding is in addition to 8B/10B; it does not replace 8B/10B).

4B/5B Corner cases and AIF2 handling:

- SSD#1 and SSD#2 form a pair that identify SOP
  - They are a unique code which give comma alignment
- ESD#1 and ESD#2 form a pair that identify EOP
  - Cannot be used for comma alignment
- Any non-data byte (except ESD) after an SSD
  - Bad Packet
  - Marked as bad packet in Multicore Navigator header (but still DMA'ed)
  - AIF2 will EOP packet immediately
- SerDes Byte Error during a packet
  - Marked as bad packet in Multicore Navigator header (but still DMA'ed)
- Packet too long
  - Not handled by AIF2 (handled by PA)
- Null Data
  - AIF2 will transfer "IDLE" when there is no control packet to send on a given control stream.

5.2.1.3 Interleave/Deinterleave

AIF2 allows up to four independent control streams per CPRI link. Each control stream may have parallel control packets in flight at different stages of completion.

CPRI Control Words Interleave/deinterleave

CPRI hyper-frames have exactly 256 control words. AIF2 provides a series of MMR registers (three bits per CW position) per link that map each CPRI control word position to

- Enable bit
  - 1'b0: CW is not mapped to a control stream
  - 1'b1: CW is mapped to a control stream
- Mapping bits
  - 0-3: map to Control Streams {0, 1, 2, 3}

With this LUT method of assigning CPRI control words, the user has complete flexibility to either duplicate the CPRI separation between

- {slow C&M, fast C&M, and vendor specific}
- Non-CPRI compliant usage of control works

5.2.1.4 Packet Parsing

The two key functions of packet parsing are

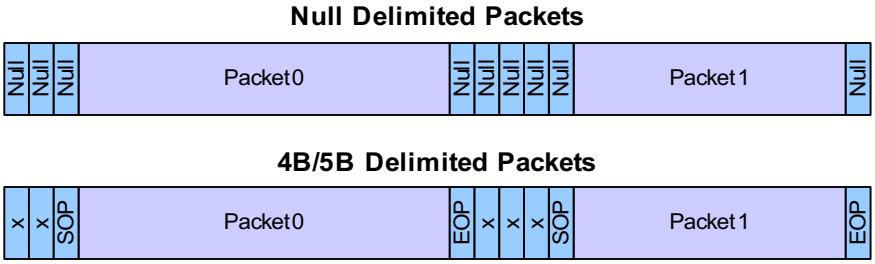
- Separating valid control data from null data
- Separating control stream into valid packets
  - Identify SOP/EOP (start/end of packet)

AIF2 has two mechanisms for packet parsing

- Programmable null delimiter
- 4B/5B coding

Each of four possible (per link) control streams is MMR configured as to which of the two options is used for that stream.

Figure 5-3. CPRI Packet Delimiters



The Programmable Null Delimiter is a unique character that identifies a null (empty byte). The value can be any byte or any non-comma K-character (e.g. K27.7 or K29.7). It is not permissible to have this character appear within a packet. AIF2 allows the user to choose which null value is to be used by programming a 9-bit MMR field. Any data that does not match the Null Delimiter is considered valid control data. There must be at least one Null Delimiter between "packets" of control data. All characters that do not contain valid control data should be filled with the chosen Null delimiter.

Alternately, the Fast Ethernet 4B/5B coding scheme can be used to identify SOP/EOP. 4B/5B may be used for non Ethernet packets. Null Delimiter scheme can not be used for Ethernet, Generic packet traffic on AxC slot and only allowed to use it for Control packet, Ethernet packet or Generic packet on Control slot.

5.2.1.5 Control Packet DMA

The AIF2 packetized control data use the Packet DMA engine for transporting and queuing of these control packets. Each possible control stream DMA channel is programmed with generic Multicore Navigator information that allows that control stream to target different device internal endpoints. Basically each ingress control stream is associated with a Multicore Navigator queue number. These particular Multicore Navigator queue numbers are associated with

- A device destination region
- PKTDMA engine (for transport)
- Second Multicore Navigator queue for action after transfer is complete

After the transfer is complete from AIF2 to the L2 (as an example), a pointer to the Multicore Navigator packet within L2 is placed on the second (destination) Multicore Navigator queue that is used to alert the CorePac of the presence (and location) of this new control data (contained in the Multicore Navigator packet).

For egress traffic, the packets for transport over AIF2 are queued in L2 (for example) where AIF2 rate controls the PKTDMA engine such that the AIF2 output buffers are not overflowed.

5.2.2 CPRI Physical Interface (Phy) Timing

SerDes Rx domains are asynchronous to the SerDes Tx domain. This retransmission operation crosses between the two clocking domains at every hop in the daisy chain. High speed SerDes are industry wide, typically four bytes wide. Each hop in the CPRI Daisy chain results in a timing uncertainty of four byte clocks.

TI has made the conscious decision to apply the OBSAI approach to link timing to the CPRI implementation. In very large daisy chains, a very large CPRI timing uncertainty would have had complex implications on several aspects of AIF2 design. (In OBSAI these link timings are referred to as Pi and Delta) AIF2 Timing section of this document shows the detail how users can set complex timing variables by setting PE1, PE2, Delta, Pi offset properly.

5.2.3 CPRI Link Maintenance {LOS, LOF, RAI, SDI}

CPRI Control Word Z.130.0 holds link management bits:

- **RAI**(b1): Remote Alarm Indicator
  - CPRI: LOS or LOF or other errors
  - TI: Programmable combination of LOS or LOF
- **SDI**(b2): SAP Defect Indictor
  - CPRI: Not fully Defined
  - TI: Set via MMR (upper layers of software)
- **LOS** (b3): Loss Of Signal
  - CPRI: 16 8B/10B code violations within a whole Hyperframe
- **LOF**(b4): Loss Of Frame
  - CPRI: FSM controlled, multiple missing K28.5 or LOS

While CPRI does define the different error conditions to some degree, CPRI does not address how to propagate these errors. The TI implementation is intended to be sufficiently programmable to handle any reasonable user requirement. In the TI implementation, each transmitted bit can be a programmable combination of:

- software set condition
- Programmable selection of source link
  - a local error condition
  - a retransmission

The LOS and LOF conditions are well defined and can occur at the receiver of any of the six links. In addition to the LOS or LOF error occurring locally, the ingress and egress CPRI link has these bits in the Z.130.0. For purposes of clarification, TI defines variant of these signals with the following subscripts:

- XXX_{RX}: Bit is received in Ingress link byte Z130.0
- XXX_{ERR}: Error has occurred locally at AIF2 receiver
- XXX_{TX}: AIF2 sets bit in Egress link byte Z130.0

At each AIF2 receiver (RM), each of the four received error bits {LOS_{RX}, LOF_{RX}, RAI_{RX}, SDI_{RX}} is received and extracted. The CorePac cores can be programmable alerted via the AIF2 EE (Error Event) mechanism. The error bits {LOS_{RX}, LOF_{RX}, RAI_{RX}} as well as the locally detected errors {LOS_{ERR}, LOF_{ERR}} are also passed to the transmitters (TM) for possible propagation. SDI is not supported with hardware propagation; SDI_{RX} is only supported via EE alert and MMR insertion that requires software support.

The transmitter (TM) for each link receives {LOS_{RX}, LOF_{RX}, RAI_{RX}, LOS_{ERR}, LOF_{ERR}} signals from each of six links as well as MMR control information and manual set of each transmitted bit {LOS_{TX}, LOF_{TX}, RAI_{TX}, SDI_{TX}}

MMR (Memory Mapped Register) bits are used to configure the usage of these {XXX_{RX}, XXX_{ERR}}bits as well as the possibility that bits may be manually set by software. Software set bits are not frame synchronized and will likely result in multiple sequential frames of XXX_{TX} signals being set on transmit.

6:1 Muxes select which of six links should be selected for generating XXX_{TX} signals. LOS and LOF input signals are grouped together as they are always relating to the same RM link. The use of these signals in XXX_{TX} signal generation is enabled by MMR bits.

LOS_{TX} is the logical OR of LOS_{ERR} OR LOS_{RX} OR an MMR bit. Each of the possible three inputs is maskable via MMR bits. LOF_{TX} is likewise comprised of {LOS_{ERR}, LOS_{RX}, an MMR bit}.

RAI_{TX} is defined as the aggregate of LOS OR LOF with the capability propagating. This is accomplished by the logic OR of {LOS_{ERR}, LOF_{ERR}, RAI_{RX}, an MMR bit}. Each possible input into the OR operation is MMR enabled/disabled.

With this mechanism, the AIF2 has tremendous flexibility in {LOS, LOF, RAI, SDI} generation and forwarding.

5.2.4 CPRI AIF2 Timer Startup

The CPRI specification implies that different nodes in the daisy chain can be either timing masters or timing slaves, but does not specify how timing information originates in the Master. Slaves are simply to use the link timing and are to pass their link timing (with as little uncertainty as possible) to the next slave in the daisy chain.

The timing circuit in the AIF2 is the AT (AIF2 Timer); It is used for both CPRI and OBSAI modes. The AT has two basic timers:

- Link (Phy) Timer {CPRI, OBSAI}
- Radio Timer {WCDMA, LTE, TD-SCDMA, WiMax.}

AT Clock Source:

- SerDes Tx Dual_ByteClk
  - {2x, 4x, 8x} 245.7MHz
  - {5x} 307.2MHz

AT (CPRI) Frame Sync Start (FSS) Sources:

- FSS device input pin
- FSS from Phy (RM)
- FSS MMR (software)

CPRI Timing Master Implementation

1. Software writes the initial value of AT timers.
2. FSS pulse starts the AT.
3. AIF2 Link transmission can begin (on next 10ms boundary)
   (a) AT provides a start strobe to TxMAC
   (b) AT provides a BFN to TxMAC
   (c) Software enables TxMAC, TxMAC starts on next 10ms link frame boundary

CPRI Timing Slave Implementation

1. AT is Off
2. RxMAC is started (Phy only w/o DMA)
   (a) receive link is up and running
   (b) RxMAC captures received BFN from CPRI Phy
3. Software qualifies that the link is in a “good” state
4. Software reads RxMAC BFN, Writes AT starting timer values
   (a) Link Timer: initial value is BFN + 1
   (b) Radio Timer: starting value can be quite a complicated calculation for radio standards that are not nice multiples of 10 ms
5. RxMAC provides FSS strobe to AT
6. AT starts timer operation

5.3 OBSAI Specifics

5.3.1 OBSAI Standard Overview

The OBSAI interface is primarily intended for passing antenna data between the RF card and the baseboard devices. Its secondary function is for passing user specific control data between any devices.

The basic premises of the OBSAI interface are to time multiplex different streams (that is, antennas) of data on a single SerDes link. OBSAI segments each data stream into 16 byte segments and appends a simple header that is used on the receiving end to reconstruct/deinterleave the different streams.

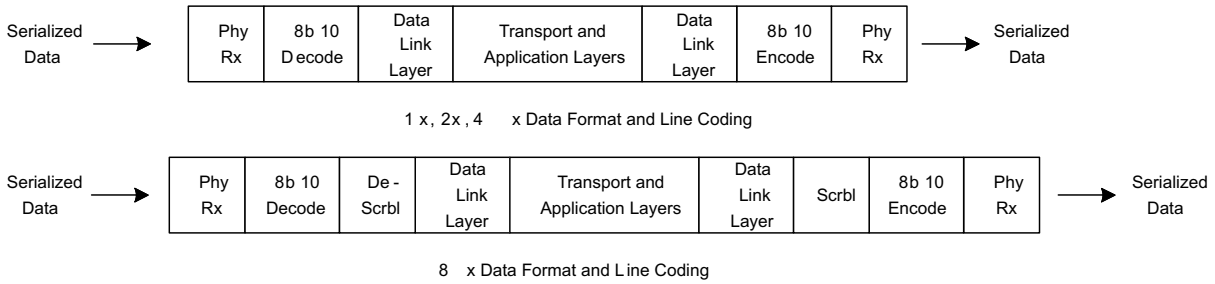
There are two basic portions to the OBSAI Interface:

- OBSAI RP1: timing interface for synchronizing all nodes/devices within the base station
- OBSAI RP3: Data Interface for passing antenna and control data between nodes in the base station utilizing SerDes connectivity.

The RP1 physical interface is a simple LVDS serial data interface with corresponding LVDS 30.72MHz clock. The AIF2 supports only reception/synchronization to RP1 sync bursts, but does not transmit RP1 sync bursts. OBSAI RP1 spec defines the serial burst stream pattern over the serial data interface and how AIF2 is to use this timing information.

The RP3 physical layer provides coding and serialization of the transmission path. An LFSR scrambling algorithm, applied to 8x links only, provides smoothing of the data stream before 8b10b encoding. The data link layer provides a method for creating messages of the bit stream. The transport layer utilizes the address field of messages to control message routing for processing of the application layer. The application layer terminates the payload of messages into packets.

Figure 5-4. OBSAI Data Format and Line Coding



The OBSAI RP3 standard supports multiple link data rates. SerDes data rates differ from usable data rates for the following reasons:

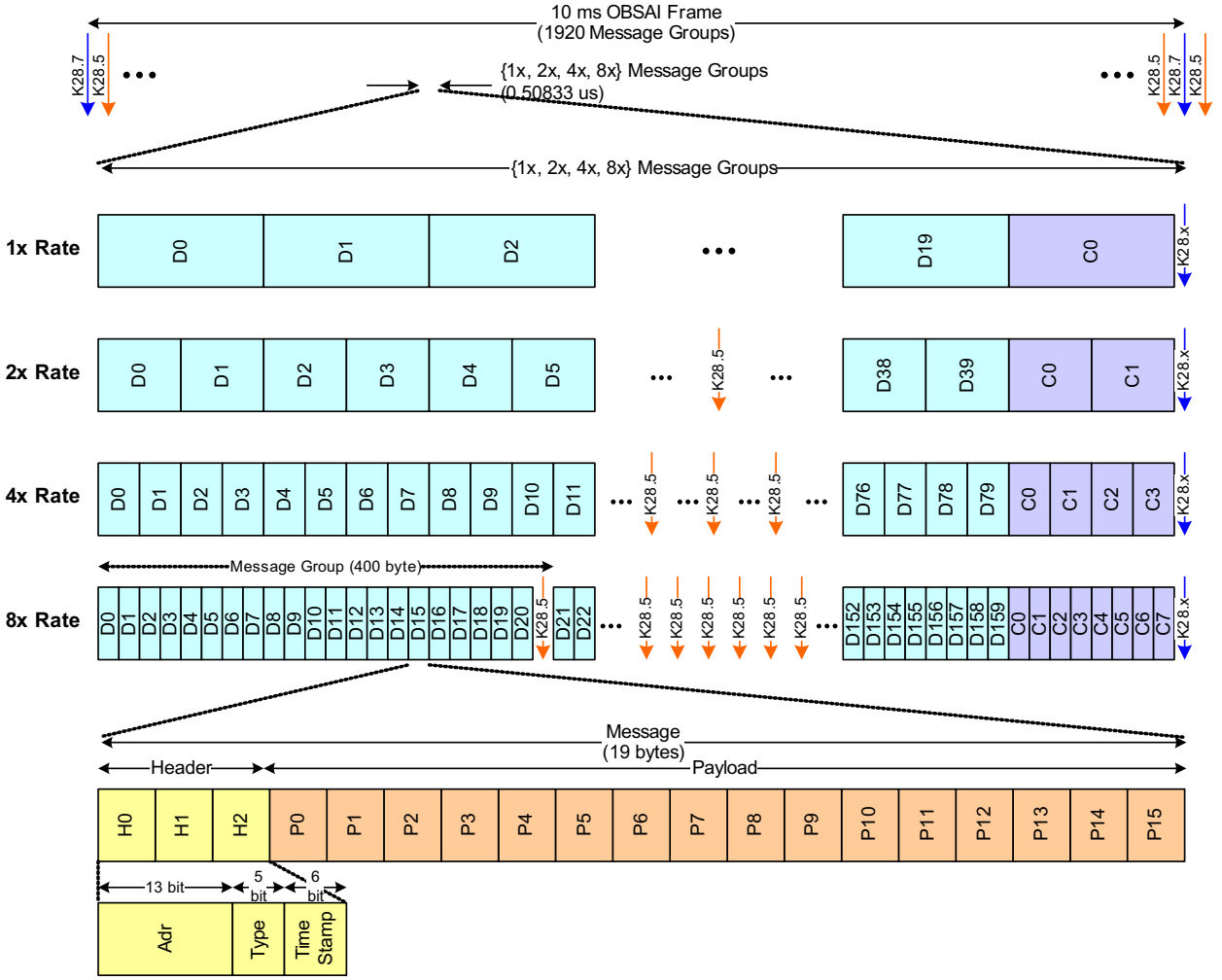
- Three bytes of header for each 16 bytes of payload
- 8B/10B SerDes encoding
- One dedicated control message for each 20 data messages
- K-character demarcating framing

Table 5-3. OBSAI RP3 SerDes Rates

Link rate	Line rate (Gbps)	Data Msg Payload rate (Gbps)	Control Msg Payload rate (Gbps)
2x	1.536	0.98304	0.049152
4x	3.072	1.96608	0.098304
8x	6.144	3.93216	0.196608

5.3.1.1 OBSAI Overview: Frame/Message Structure

Figure 5-5. OBSAI Frame/Message Structure



Regardless of link rate, OBSAI has a 10 ms frame structure. The frame boundaries are demarcated with a K28.7 character at the end of each frame. Each frame is further subdivided into *i*1920 message groups (where *i* depends on link rate *i*={1x, 2x, 4x, 8x}). Each message group is exactly 400 bytes containing exactly 21 messages ending with a single K-Character:

- K28.7 if it is the last message group in a frame
- K28.5 for other messages groups in a frame

Data Messages are grouped with Control Messages. First there are *i*20 Data messages then *i* Control Messages. These groups span multiple message groups (with K-characters ending each message group.

Table 5-4. OBSAI Data/Control Message Grouping

Link rate	i Message Groups	Data Messages	Control Messages
2x	2	40	2
4x	4	80	4
8x	8	160	8

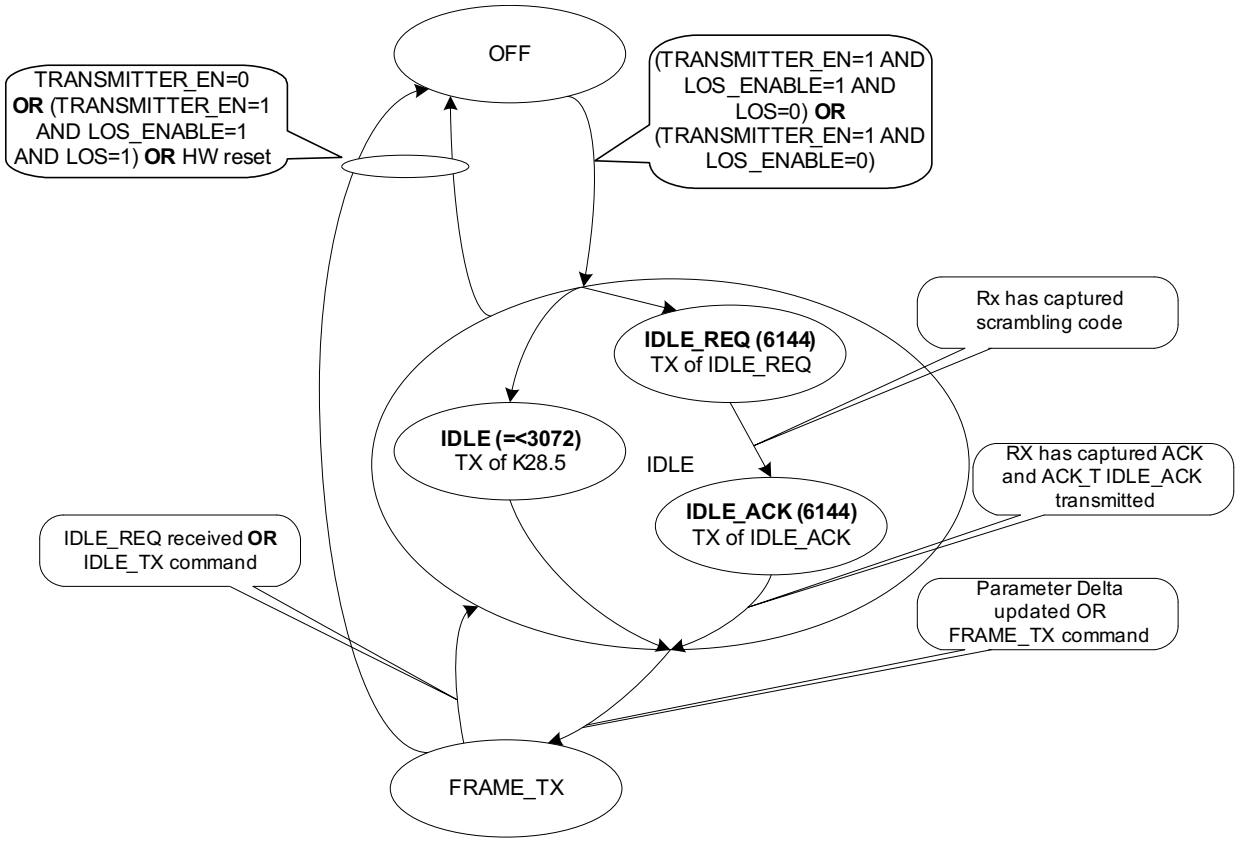
The special message type of “Empty Message” is defined to indicate that a control or data message is currently unused. “Empty Message” is indicated with the special OBSAI address of 13’h1fff. All other fields are “don’t care” but AIF2 will fill all other fields with zeros.

Figure 5-6. OBSAI Empty Message (AIF2 Transmitted)

5.3.1.2 OBSAI Overview: RP3 Tx and Rx FSMs

Frame synchronization is achieved by adhering to defined state transitions in OBSAI RP3. For transmit, the state transitions are shown in Figure 5-7. The frame synchronization begins in the Off state. The transmitter is enabled by system control to transition the state machine to the Idle state. While in the Idle state, the state machine follows the scrambler protocol acknowledge sequence if the link is configured for 6144 Mbps. If the link is configured for less than 6144 Mbps, then while in the Idle state, K28.5 characters are transmitted. Once the delta variable is known, the state machine can be transitioned to the FRAME_TX state, and valid frames are transmitted at the delta offset. Upon a reset, receiver loss of signal, or a system control disable of the transmitter, the state machines transitions back to the Off state.

Figure 5-7. OBSAI RP3 Transmit State Machine



The receiver state machine in Figure 5-8 consists of four states for 768, 1536, and 3070 Mbps line rates and six states for the 6144 Mbps line rate. The four states applied to all line rates are as follows:

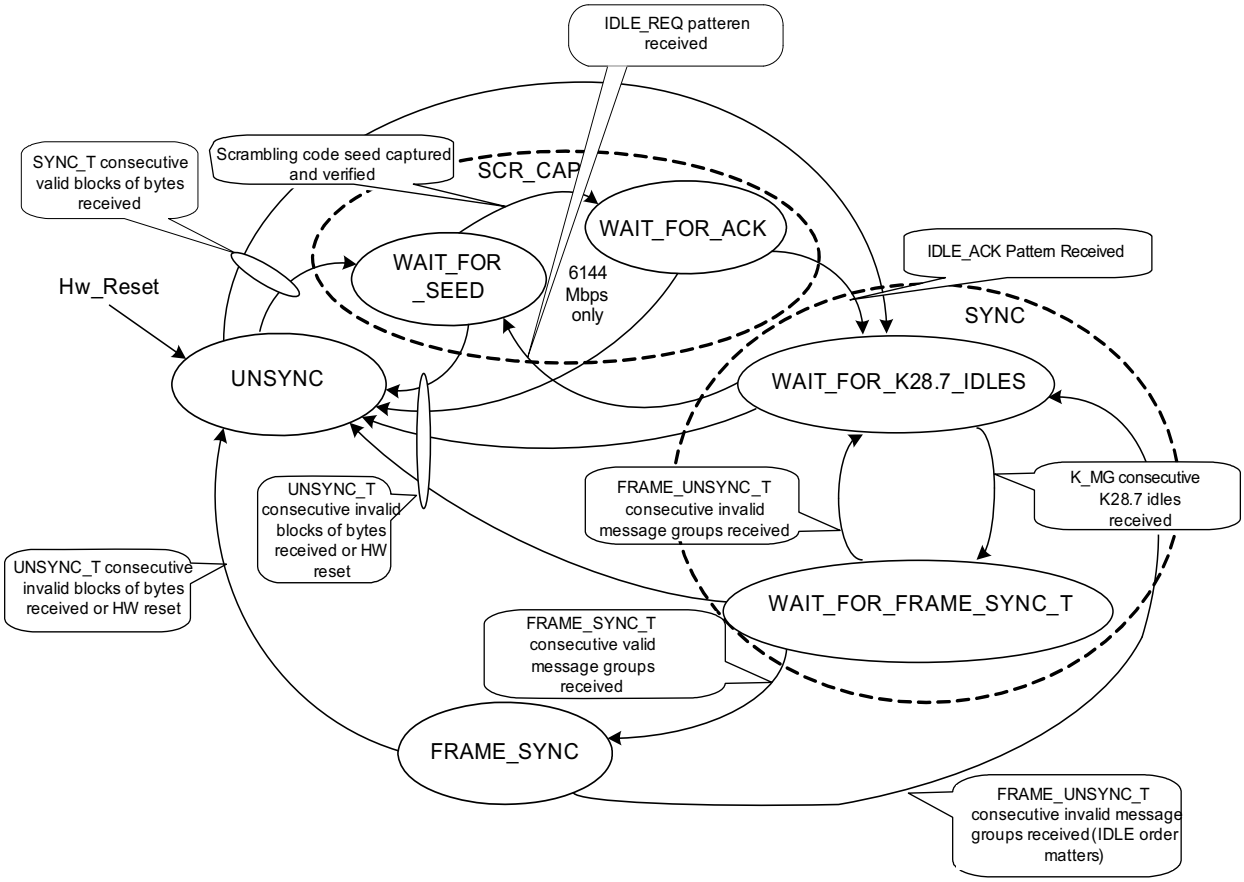
- UNSYNC
- WAIT_FOR_K28.7_IDLES
- WAIT_FOR_FRAME_SYNC_T
- FRAME_SYNC

Two of these states, WAIT_FOR_K28.7_IDLES and WAIT_FOR_FRAME_SYNC_T, can be considered to form a single logical state called SYNC. The meaning of states UNSYNC, SYNC, and FRAME_SYNC is as follows:

- UNSYNC: Bus link is down. Many byte errors are detected.
- SYNC: Bus link is working, that is, connection exists.
- FRAME_SYNC: Normal operational mode. Frame structure is detected and messages are received.

Two additional states are applied for the 6144 Mbps line rate due to scrambling seed transfer from the transmitter to the receiver: WAIT_FOR_SEED and WAIT_FOR_ACK. These two states form a logical state called SCR_CAP (scrambling seed capture). It is expected, that the receiver can capture scrambling code both from IDLE_REQ and IDLE_ACK patterns. When in the state WAIT_FOR_K28.7_IDLES, IDLE_REQ is detected if every 17th byte of received data is a K28.5 and there are valid data bytes (no K-codes, no LCV errors) between K-codes. Contents of data bytes are not checked. This way it is possible to recognize IDLE_REQ, even if the scrambling code has been changed.

Figure 5-8. OBSAI RP3 Receive State Machine



The receiver synchronization state machine has several outputs. Current state of the receiver will be available for Application layer. An interrupt may be generated from each state change. Signal loss_of_signal is active (has value 1) in state UNSYNC while it is inactive in other states. Synchronization block also indicates bytes that contain 8b10b decoding error as well as the location of the Master Frame boundary. For the 6144 Mbps line rate, scramble seed capture and acknowledge training pattern received is also indicated.

5.3.1.2.1 OBSAI Overview: General

Streams/Messages are classified into different types, and these types are indicated by the type field of the message headers.

Table 5-5. OBSAI RP3 Types

Payload data Type	Content of Type Field	Timestamp #: address extension X: Timestamp Protocol
Control	00000	000000 –or- 000001
Measurement	00001	XXXXXX
WCDMA/FDD	00010	XXXXXX
WCDMA/TDD	00011	XXXXXX
GSM/EDGE	00100	XXXXXX
TETRA	00101	undefined
CDMA2000	00110	XXXXXX
WLAN	00111	undefined
LOOPBACK	01000	undefined
Frame Clock Burst	01001	000000
Ethernet	01010	XXXXXX
RTT message	01011	000000
802.16	01100	XXXXXX
Virtual HW reset	01101	000000
LTE	01110	XXXXXX
Generic Packet	01111	XX####
Not Used	10000-11111	undefined

OBSAI supports several different types of traffic. These are identified by the “Type” field in the OBSAI header where the different types are predefined as a five-bit encoded fields. For the different types, the usage of the “Timestamp” fields varies. Table 5-5 shows the “Timestamp” usage for the different OBSAI types. An “X” indicates that there is an algorithm for filling the Timestamp. A “#” indicates that Timestamp bits are being used to extend the addressing. In other cases, a fix value is written.

CRC error check is provided for the following types:

- Control Messages: 16-bit CRC
- Generic Packet Type: 16-bit CRC
- Ethernet Type: 32-bit CRC

5.3.1.3 OBSAI Overview: Timestamp

The OBSAI header has a six-bit field that is labeled Timestamp. For transport types that are streaming antenna traffic, this timestamp fields is used to indicate the progression of samples moving in/out of the RF card. Additionally, the TS fields carries framing information indicating radio frame boundary for all standards except for GSM/Edge where timeslot boundary is indicated.

5.3.1.3.1 Non-GSM AxC Timestamp

The six bits of Timestamp start at 6’b000000 at the radio frame boundary and increment by one every OBSAI message. Timestamp is maintained separately for each AxC. The timestamp value ranges from 0-to-63, wrapping when reaching the max count. Timestamp equal to 0 can either mean radio frame boundary, or a simple wrap of the counter.

5.3.1.3.2 GSM Timestamp

GSM/Edge has a custom Timestamp where the MSB represent the beginning of a GSM timeslot (1'b1 identified timeslot beginning). The remaining LSBs start at zero (at the beginning of a timeslot) and increment one for each message of the timeslot. There is one slight difference between UL and DL:

- DL: The MSB is held high for only the first OBASI message of the timeslot
- UL: The MSB is held high for the first four OBSAI messages of the timeslot

5.3.1.3.3 Ethernet Timestamp

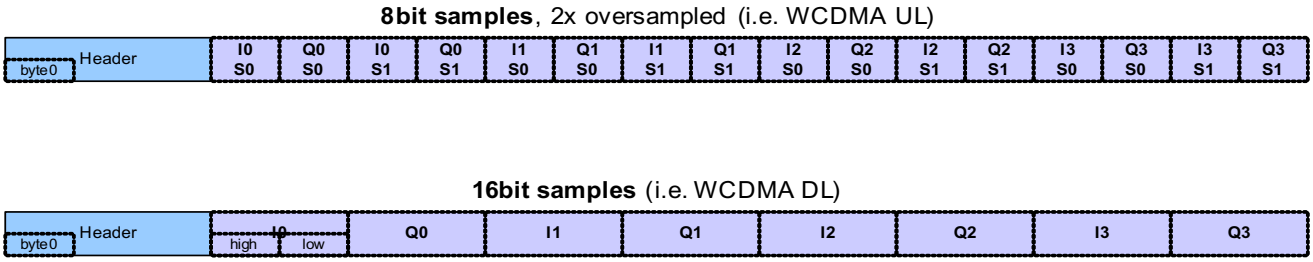
- SOP: 6'b100000
- MOP: 6'b000000
- EOP: 6'b1XXXXX (XXXXX: indicates the number of bytes from the start of RP3 payload containing MAC frame data (counting started from the byte after the header)

5.3.1.3.4 Generic Packet Timestamp

- SOP: 6'b10XXXX
- MOP: 6'b00XXXX
- EOP: 6'b11XXXX (XXXX: is an extension of OBSAI address and is the same for all elements within the packet)

5.3.1.4 OBSAI Overview: Message Payload Formats

Figure 5-9. OBSAI Message, Payload Sample Packing



Two different payload formats are supported for antenna traffic:

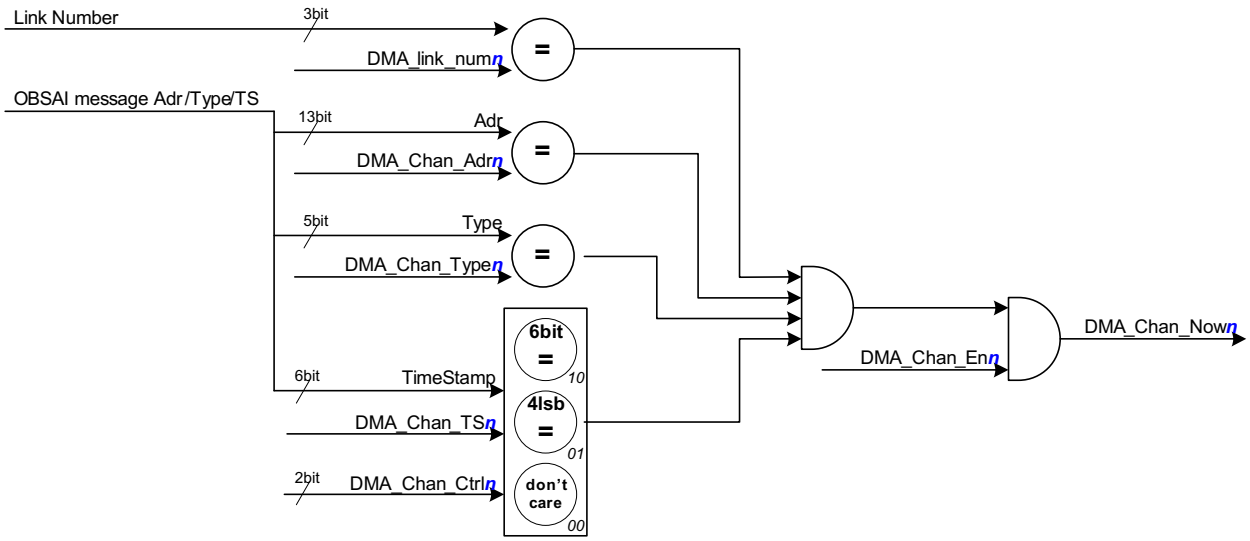
- 8-bit I + 8-bit Q, 2x over-sampled
- 16-bit I + 16-bit Q, not over-sampled

See Figure 5-9 for data alignment within the OBSAI message payload.

5.3.2 OBSAI: Header Based Processing

5.3.2.1 DMA Channel Index

Figure 5-10. OBSAI DMA Index Circuit (Header-Based Reception)



OBSAI Header information is used to map ingress stream data with the appropriate Ingress DMA channel. There are a total of 128 possible Ingress DMA channels per AIF2. The total address space of an OBSAI header is too large to implement with a exhaustive RAM LUT, so (effectively) a 128 deep CAM (Content addressable Memory) is built. 128 parallel comparison circuits are built corresponding to the 128 possible ingress DMA channels. When one of these 128 comparison circuits activates, the corresponding DMA channel is indexed.

NOTE: It is the user's responsibility to insure that each unique DMA channel LUT is programmed uniquely such that only one comparison will fire at a time.

Mostly this indexing function is based on the OBSAI Address and Type comparison. The exception to this rule is the OBSAI Generic Packet indicated by type 5'b01111; The address space is effectively extended by using the four LSBs of the Timestamp field in Generic Packet Mode. In Generic Packet Mode, there is not valid concept of timestamp and so this field is used to extend addressing.

While AIF2 has 128 different DMA channels allowing for unique handling and destination addressing of all 128 streams, AIF2 allows for multiple OBSAI Addresses to map to a single PKTDMA channel. Control packets, being only 16 bytes wide, are the best candidate for this usage, as it avoids the problem of two different partially composed packets to collide on the same DMA channel (which is illegal).

5.3.2.2 8B/10B Code Violation and Bad OBSAI Message Header

A SerDes bit error is expected to occur one every 10^{15} bits (results in an 8B/10B code violation). Although infrequent, it happens regularly enough that it needs to be handled properly. Each 10 bits across the SerDes map to an eight-bit byte; a single bit error results in the complete corruption of the entire byte.

For antenna traffic, an infrequent bad byte at the phy level has much lower error contribution than the cellular air channel. For control data, a bad byte in the payload fields is assumed to be unimportant, caught by CRC check, or corrected by higher layers. The AIF2 handling of bad bytes in the OBSAI payload (of the message) is to convert it to a 0 (8'h00) and simply pass it along.

5.3.2.2.1 AIF2 Operation

- The AIF2 RM (Rx Mac) drops packets with:
  - 8B/10B code violations in the OBSAI Header –or-
  - K30.7 in the OBSAI Header
  - Dropped packet is replaced with empty message
- The AIF2 RM will replace bad payload bytes with 8'h00. Bad byte is identified by either
  - 8B/10B code violations in the OBSAI Payload –or-
  - K30.7 in the OBSAI Payload

Retransmission occurs after RM, so the above actions dictate the handling

5.3.2.3 OBSAI: Missing Timestamp

AIF2 expects a constant stream of correct OBSAI messages for each AxC stream. AIF2 checks the incoming stream by examining the OBSAI message header timestamp field. Timestamps are expected to be zero on radio frame boundary and increment by one for each OBSAI message of a given AxC.

Due to the handling of 8B/10B code violations in OBSAI headers, whole OBSAI messages can be dropped. AIF2 has handling mechanisms that tolerate one or two consecutive missing messages. If there are more than two consecutive missing messages, the entire AxC is failed; resynchronization occurs on the next radio frame boundary.

5.3.2.3.1 AIF2 Operation

- PD Performs TS check
  - PD predicts TS and FB
  - Checks TS against Expected
  - Writes packet boundaries and Error signals into FIFO w/ Data
- DMA interface block (while reading DMA Channel FIFO)
  - Insert 0, one, or two extra empty data phases
  - Close/Open packet as indicated by packet boundaries

The AIF2 partitions the handling of missing TS information among several submodules. The PD (Protocol Decoder) writes whole messages into the DB FIFO and identifies missing timestamps and missing packet boundaries. PD performs this check using predicted timestamps and symbol boundaries.

PD passes a failed packet signal along so that AD can drop the packet through the PKTDMA engine. PD can fail a packet either because of missing messages, or because of a failed CRC check. PD supplies packet boundary (SOP) and EOP indicators through the DB FIFO to AD for the purpose of segmenting the data stream into packets. The DB (Data Buffer) FIFOs simply pass and flow control the data to the AD (AIF2 DMA engine).

The AD inserts zeroed data and recreates missing data so that the total number of data bytes DMA'ed will be the same as if no messages were missed at all. AD inserts one or two empty messages worth of traffic depending on how many messages were missing. In the event that one of these messages would have been a packet boundary (such as LTE Symbol boundary), AD closes the current packet appropriately, and opens a new packet appropriately. In the event that an error occurs, it is traced in the EE mechanism.

These mechanisms can handle at most two missing messages per stream. In the event that there is an extra message, or more than two missing messages, PD will "Fail" the AxC.

5.3.2.3.2 Failed AxC

- PD restarts the TS predictor circuit
- PD close out any open packet (by writing a EOP with zero data)
- PD will activate Fail_Pkt signal marking packet as bad
- DB operates like normal as if no failure occurred
- AD
  - Direct IO: PD stops capturing data until next Radio Frame Boundary; Direct IO DMA continues to transfer junk.
  - Multicore Navigator: Discard current packet for DMA channel

There is a watch dog timer per PD channel used for terminating GSM packets in the event an OBSAI message is lost and that message was the EOP. The feature is generic and can be used in any radio standard, but not with other packet types such as Generic Packet mode. The user programs a maximum number of clock cycles expected between OBSAI message arrivals (which should approximate a form of the sampling rate for the radio standard). In the event that a message is not received in the expected time window, and an EOP was expected, then the packet is terminated. (This circuit will not handle the case of two missing OBSAI messages where the second message is the EOP).

5.3.3 OBSAI: 6 GHz Scrambling

Bit level scrambling is performed on 8x rate links to reduce cross talk between links as well as to reduce inter-symbol interference (ISI). The RP3 transmitter applies a seven-degree polynomial to data bytes and the inverse operation is performed by the RP3 receiver. Scrambling only pertains to 6 GHz operation (8x link rate). Link rates {2x, 4x} are backward compatible with no scrambling applied.

Cross talk between transmitters through the local SerDes power supply is the main concern. With all transmitters having differing scrambling offsets, randomness between transmitting lanes is achieved. The assignment of unique scrambler offsets for receivers is optional as cross talk between receivers and transmitters is non-critical.

The RP3 transmitter is configured by higher layers with a starting value of the seven-degree polynomial scrambling code generator. Higher layers should configure unique seed values for adjacent RP3 Tx links. Table 5-6 lists the available seed values to be used; these seed values represent nx7 position offsets (nx7 has been specifically chosen to give an odd offset between adjacent links). The RP3 receiver is a slave to the transmitter, receiving the seed value in a training sequence.

Table 5-6. Scrambler Seed Values

Nx7 Index	X^7	X^6	X^5	X^4	X^3	X^2	X^1
0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	1
2	0	0	0	0	1	0	1
3	0	0	0	1	1	1	1
4	0	0	1	0	0	0	1
5	0	1	1	0	0	1	1
6	1	0	1	0	1	0	0
7	1	1	1	1	1	0	1
8	0	0	0	0	1	1	1
9	0	0	0	1	0	0	1
10	0	0	1	1	0	1	1
11	0	1	0	1	1	0	1
12	1	1	1	0	1	1	0
13	0	0	1	1	0	1	0
14	0	1	0	1	1	1	0
15	1	1	1	0	0	1	1
16	0	0	1	0	1	0	1
17	0	1	1	1	1	1	1

The scrambler is a seven-degree polynomial, linear feedback shift register (LFSR). The polynomial is $X^7 + X^6 + 1$. K28.5 or K28.7 characters reset the LFSR to the seed value. The bit pattern repeats every 127 bits.

5.3.4 OBSAI: Generic Packet Mode

Generic Packet is an OBSAI mechanism that allows multiple OBSAI messages to be grouped together to form a larger packet. The packet size is n x 16 bytes where the last two bytes are the CRC16.

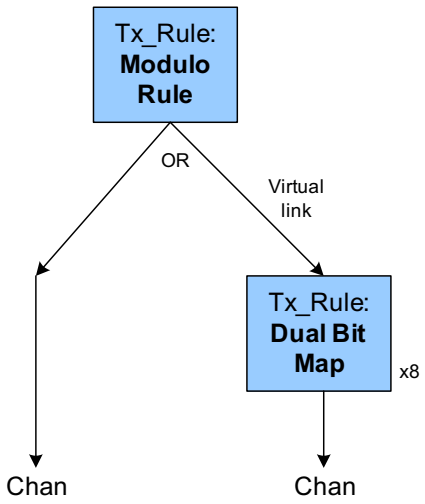
Generic Packet

- Type fields = “5’b01111”
- Timestamp field
  - Bit[5:4]
    - 2’b10: SOP start
    - 2’b00: MOP mid-point
    - 2’b11: EOP end
  - Bit[3:0]: extension of address field

5.3.5 OBSAI: RP3 Transmission

Modulo and Dual Bitmap Rules are used in conjunction with each other to flexibly allocate the Link bandwidth between multiple streams of antenna or control data.

Figure 5-11. OBSAI Transmission Rules



5.3.5.1 OBSAI Tx: Modulo Rules

Data Messages and Control Messages are separated into two different control flows. A counter is maintained for each where the counter is zeroed at the beginning of each PHY (RP3) frame boundary. A modulo value is chosen and different streams are given a unique indexes into that modulo.

The modulo technique is used to share bandwidth between different streams sharing the same link or virtual link. It is used for either antenna traffic or control traffic, but it was not intended that a single stream be shared between Control Message slots and Data Message slots.

The Modulo technique works very well for WCDMA, LTE, and TD-SCDMA where the sample rate is an exact divisible rate relative to the OBSAI Data Message rate. The Modulo technique does not work for WiMax where the sample rate does not have a clear relationship to the OBSAI rates.

Normally, TI recommends to use two Modulo rules, one for AxC streams and the other for Control streams and each modulo rule could be matched same sequence of DBM rule. Chapter 9 shows more details about how to set up rules for several cases.

5.3.5.2 OBSAI Tx: Dual-Bit Map Rules

The Dual Bit Map approach is a rate matching algorithm for mapping steams into links (with dissimilar frequencies). In summary, *X* AxC with the same sampling frequencies are mapped into *X* sequential Data Messages. At the end of *X* sequential Data Messages one empty message is inserted according to the programming of the Dual Bit Map control fields. Dual Bit Map Rules also could be used for Control Messages or Generic packet with Modulo rule if the user does not use any bubble slots.

If the bit map contains:

- 1: length = *X* + 1 (*X* data message for *X* AxC + one Empty message)
- 0: length = *X*

The index into the bit map is incremented each X or X+1 data messages. The empty message gaps constitute the rate matching where the whole purpose of the Bit Map Algorithm is to place these empty messages in meaningful locations.

OBSAI RP3 specifies programming values of the Dual Bit Map FSM in the RP3 specification, Appendices D and F.

AIF2 DMA (Multicore Navigator and DIO) Information

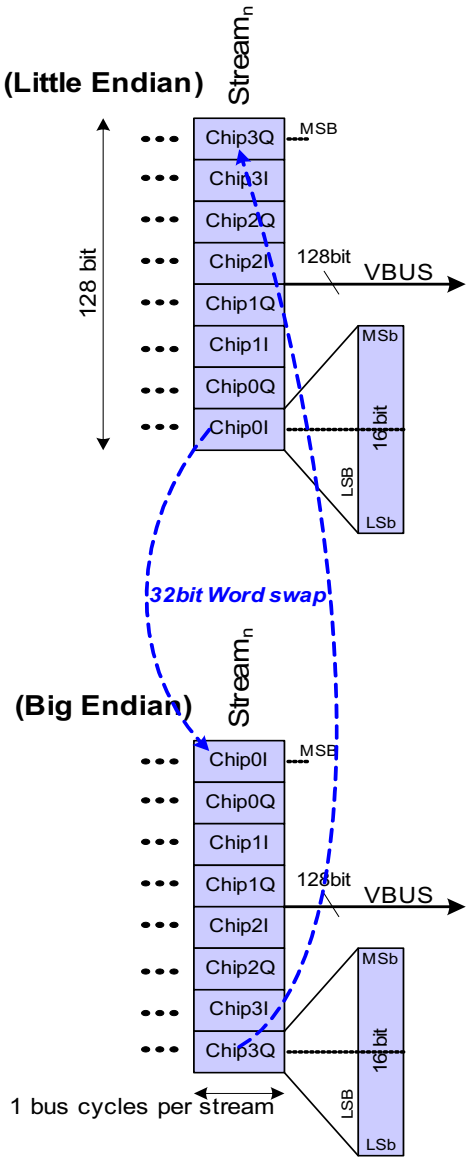
Topic **Page**

6.1 VBUS Data Formats 105
6.2 AIF2 Direct IO 108
6.3 PKTDMA 111
6.4 Multicore Navigator Examples 117

6.1 VBUS Data Formats

6.1.1 AIF2 16-Bit Sample DMA (OFDM and WCDMA DL)

Figure 6-1. 16-bit Sample VBUS Transfer Format



Four samples of 32 bits each conveniently fit into a single 128-bit VBUS cycle. For this reason, VBUS packing and AIF2 DB FIFO packing are orientated as 128-bit (eight 16-bit values).

Endianess handling is performed on the 32-bit entity size where word alignment swapping is implemented.

This data format is used for

- WCDMA DL
- LTE, WiMax, TD-SCDMA

Figure 6-1 shows a WCDMA chip, but the same figure applies to OFDMA standards by replacing the word “chip” with “sample”.

In many cases, CPRI data arrives as 15-bit I + 15-bit Q. The Protocol layer converts between 15-bit and 16-bit data prior to the DMA layer.

WCDMA DL data has two possible sources:

- TAC
- L2 (RSA processed DL data)

6.1.2 WCDMA UL DMA

Figure 6-2. Uplink VBUS Transfer Format

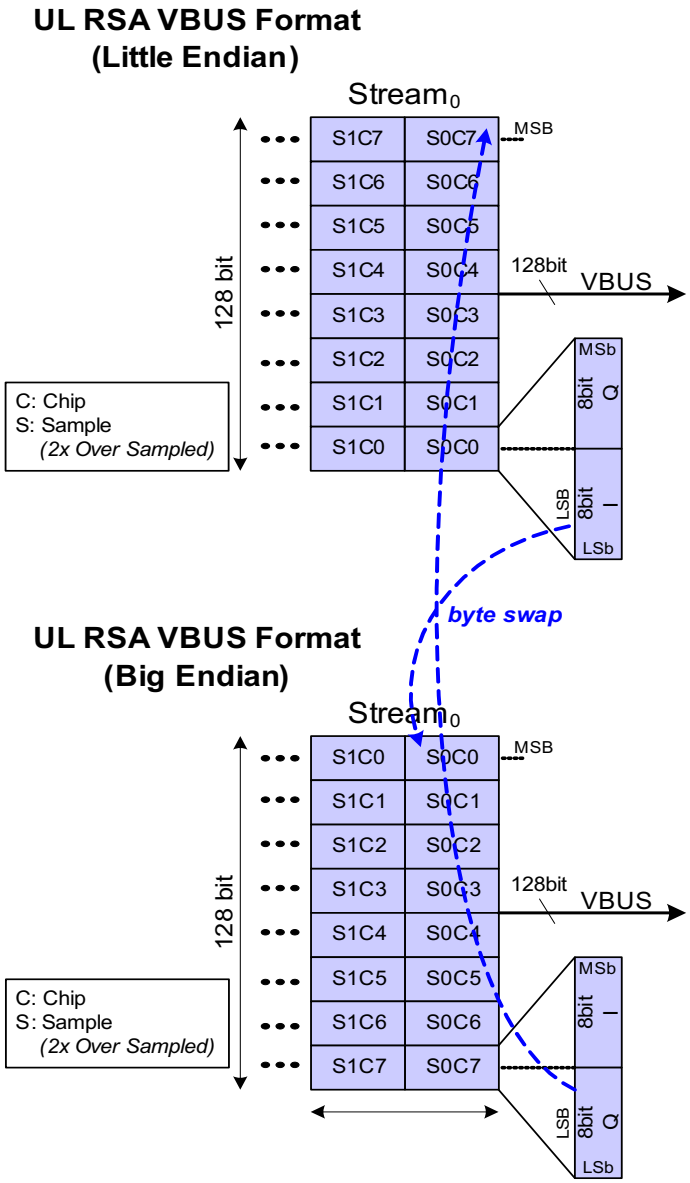


Figure 6-2 shows:

- Odd samples: S0xx
- Even Samples: S1xx

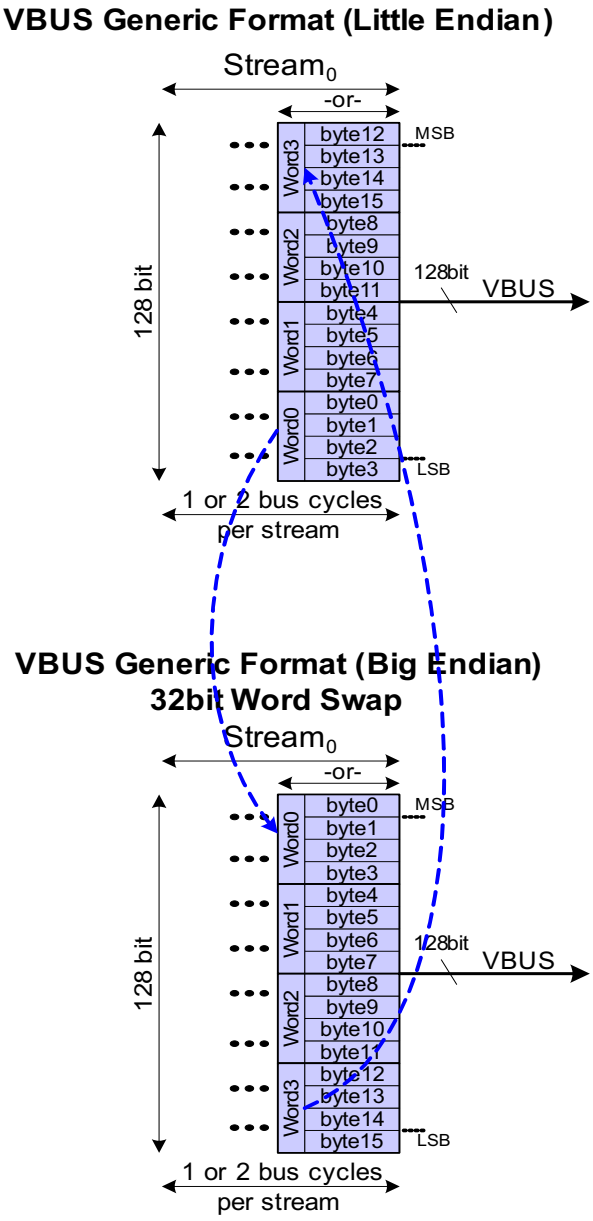
RSA Data format is shown in diagrams. This RSA data format is the WCDMA UL data format used by both CorePac RSA and RAC. The figure shows Odd/Even samples sequentially transferred over the VBUS (in separate data phases). This is the way the RAC FEI requires input data.

The uplink data flow is somewhat more complex than the Downlink scenario. Predominately, Uplink symbol data originates in the Inbound RAM and is written to both the RAC (Receive Accelerator Coprocessor) and the RX DSP L2 RAM. Delayed streams are formed by writing Antenna Data to the external RAM and then retrieving it at a later time. Retrieved data is sent both to the RAC for receive processing and out the Antenna Interface for broadcast to other devices in the system.

WCDMA Uplink DMA transfers are always bursts of eight chips *(the DL transfer burst format is four chips)*. Uplink data is eight bits I and eight bits Q with an over sampling rate of 2x. The RSA (which can perform some Uplink processing such as Preamble Detect or Path Monitoring) requires the over sampled input data to be separated into even and odd samples. All this coupled with a VBUS width of 128 dictates an optimal transfer burst of eight chips.

6.1.3 Generic Format DMA

Figure 6-3. VBUS Format Generic (Byte Numbers Relative to PHY Arrival)



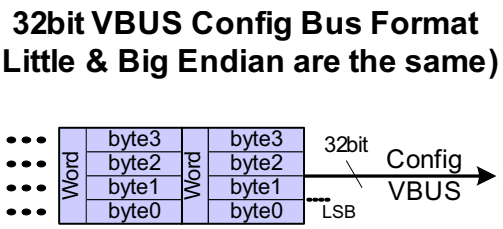
Generic Format:

- CPRI Ethernet
- CPRI Control Data
- OBSAI 16 byte (mode) Control Message
- OBSAI or CPRI Generic Packet of Control Slot
- OBSAI or CPRI Generic usage of AxC Data Payload

The Generic format is envisioned to support all packet switch messages (including control messages) and is used to facilitate inter-device communication traffic. The Generic format is differentiated from other formats with its word (four-byte) orientation matching the width of the DSP core data path. (This width assumption only has an impact on Big Endian mode of operation.)

6.1.4 VBUS Config Bus

Figure 6-4. MMR VBUS_Config Data Format



The 32-bit VBUS config port is used for reading and writing MMRs within AIF2 as well as accessing the AIF2 data path for emulation.

All MMRs within AIF2 are defined as 32-bit words and are therefore have identical Big Endian and Little Endian representation on the 32-bit VBUS_config.

While users are debugging code using Code Composer Studio, it is possible to place various memories into a watch window. To achieve this, CCS is actually performing reads (and writes) of various memory mapped locations. AIF2 gives access to the DB through the VBUS config interface.

6.2 AIF2 Direct IO

The term Direct IO means that a peripheral has dedicated custom logic that implements data movement (as apposed to using EDMA or CPU reads/writes). For AIF2, custom circuitry is built to handle data movement requirements unique to WCDMA.

The Packet DMA (PKTDMA) module has a Direct IO support feature that allows AIF2 to pass VBUS reads/writes to the VBUS, using the 128-bit VBUSM master port on the PKTDMA. The AIF2 CPPI_Sceduler gives Direct IO accesses higher priority than Multicore Navigator packet transfers.

Direct IO is a state machine that is triggered to transfer data when internal AT system events fire. Example transfer times could be every {4, 8, 16} chips of time. The time granularity of the data transfer does depend on UL/DL and the preferred packing at the destination (that is, DDR3 prefers 64 byte read/write bursts). The amount of data to be transferred is equal to the accrued number of sample buffered during the transfer time chosen above.

The PKTDMA module in AIF2 megamodule assigned 128 channels (channel number 0 ~ 127) for Packet style data transfer like LTE, WiMAX, TD-SCDMA, GSM and also assigned channel number 128 for Direct IO purpose. The PKTDMA Rx and Tx Channel 128 should be enabled before sending or receiving DIO data and PKTDMA internal data loopback mode also should be disabled because enabling loopback is the default value for PKTDMA. Users can see more detail about PKTDMA setup for DirectIO from Multicore Navigator User's guide

6.2.1 AIF2 Direct IO

- Circular Buffers
  - 32 –or- 64 chips (128 or 256 bytes)
  - Offset into buffer is a function of DIO offset
- Ingress Destinations
  - RAC: UL data
  - DDR3: UL data for delay
  - L2: UL data for RSA processing
- Egress Source
  - TAC: DL data
  - L2: RSA generated DL data
- Direct IO of WCDMA data
  - Triggered by AT internal event
  - Three different interfaces for UL (three different possible destinations)
  - Two different interfaces for DL
  - One interface for Data Trace
  - Resynchronization
  - Circular Buffer Depth : 32 or 64 chip Wrap
  - DDR3, Burst of two AxC to get Burst length up to 64 byte

Direct IO is legacy support for WCDMA traffic. Because of the nature of WCDMA, the DMA is more circular buffering in nature particularly with the legacy support of the RAC hardware accelerator module that has a predefined interface. The concept of a FIFO for WCDMA data just does not work very well.

When reading or writing to DDR3, it is important that the data burst be 64 bytes or larger and addresses be aligned on 64-byte boundaries. Failure to adhere to this will result in poor DDR3 performance (consume two times the DDR bandwidth and require more clock cycles).

This Direct IO support uses the DB RAM that was otherwise used to build FIFOs.

Direct IO supports two AIF2 DB internal buffer sizes only:

- 128 byte: intended for WCDMA use
- 256 byte: intended for higher rate LTE uses

It is the responsibility of the programmer to allocate DB RAM consistent with the 128byte or 256byte memory option. All Direct IO channels are required to have the same depth (128 or 256 bytes). It is permitted to mix Direct IO and Multicore Navigator channels in the DB, and there are no restrictions on Multicore Navigator usage while Direct IO is used.

The programmed DIO offset represents the difference in External AxC offset relative to other antenna carriers. User can setup Ingress DIO offset in PD DMA channel config 1 register and Egress DIO offset in Egress Data Buffer Channel Configuration Register. The unit of this field is QW (quad word).

External AxC offset means the pure data delay for external line latency and DSP internal delay is not included in this calculation. If there's no external AxC line delay, it could be set to zero. The more precise formula follows:

RoundUpBy IterationSize { ***External AxC offset*** Mod ***BufferSize***}

- **IterationSize** = 4chip for DL (1QW) or 8chip for UL (2QW)
- **BufferSize** = 32 or 64 chips (8 or 16 QW)

For example, if External AxC offset is 4 chip (first incoming 4 samples of data will be junk) and BufferSize is 32 chip (8QW), then the Mod value is 4chip and the final value rounded up by IterationSize (4 chip) is also 4chip.

So user can set 1 for Ingress or Egress DIO offset and it means PD will shift Ingress DB buffer read start offset to avoid reading junk or Egress DB will insert valid data from second QW in Egress DB buffer and PE will insert first 4 chip data as junk from DB

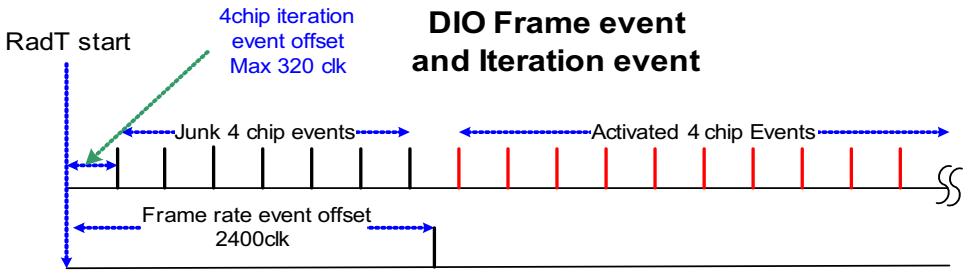
Direct IO is a precisely timed event relative to Multicore Navigator which is a data-arrival driven event. The AT creates internal system events that control the timing. Two separate system event control the Direct IO DMA

- Frame rate strobe ==> at_ad_*gr_event[3] ~ [5] for three DIO engines
- iteration strobe ==> at_ad_*gr_event[0] ~ [2] for three DIO engines
  - Four chips for TAC DL
  - Eight chips for RAC UL
  - Flexible for other burst sizes

The maximum offset size for DIO 4chip or 8chip iteration event offset (at_ad_*gr_event[0 ~ 2]) should not be bigger than event mod size (if modulo is 320 clock, event offset should be same or smaller than that) and this means user can not have bigger offset value than 4 chip or 8 chip size. Frame rate event (at_ad_*gr_event[3 ~ 5]) was created to cover this limitation. User can set very large amount of offset by using Frame rate event offset. 4chip or 8chip iteration event will not be activated until Frame rate event occurred after RadT start time.

Figure 6-5 shows how 4 chip iteration event is activated after 2400 clock of frame rate event offset. In this case, 4chip iteration event offset could be used for fine offset control to make it start right after the frame event offset

Figure 6-5. Relation Between Frame Rate Event And 4chip Iteration Event



There are three parallel Ingress Direct IO engines so that the same Ingress data can be DMA'ed to {RAC, RSA, DDR3}. There are three parallel Egress Direct IO interfaces to support the three sources of egress data {TAC, RSA, DDR3}. Only one source of Egress data will ever contribute to the same AxC; the user chooses to access which of the three different sources by programming the Direct IO.

6.2.2 AIF2 Direct IO Link Data Trace

The Data Trace is supported through a custom Direct IO mechanism. Each time 64 bytes of data is captured (and 16 bytes of associated side band data), a Direct IO transfer is scheduled.

Data Trace RAM Size:

- Byte Data: 1kbytes: 16byte x 64 deep
- Side Band: 256bytes: 16byte x 16 deep

Data Trace Direct IO has lower priority than antenna data Direct IO. For this reason, the Data Trace Buffer is chosen to have a depth of approximately eight chips of time.

DMA can be programmed to transfer (controlled by DB and RM):

- No Data Trace Data
- Only captured data bytes
- Only sideband signals (K-Char indicator and 8B/10B code violation)
- Both data bytes and sideband signals

For each burst of Data Trace DMA:

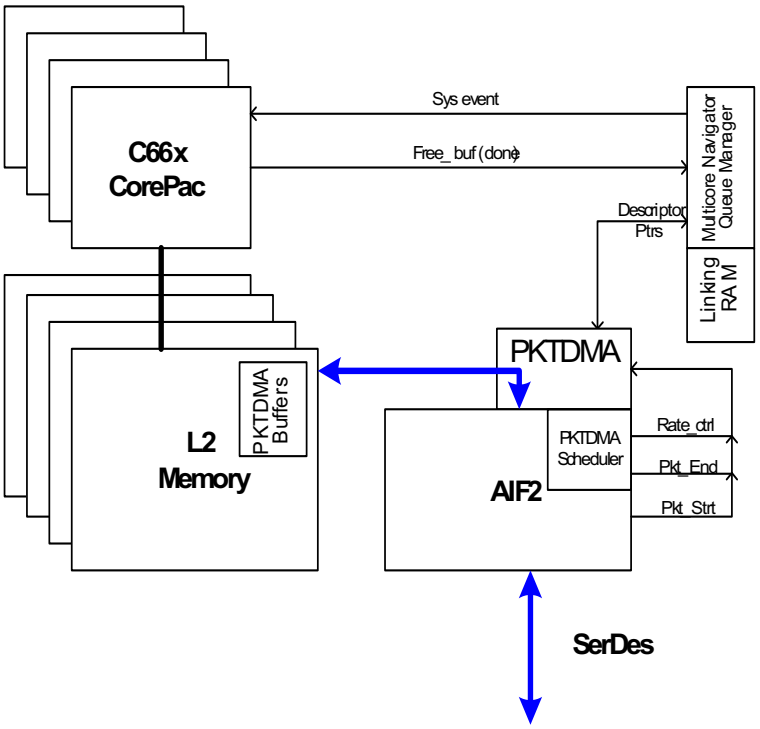
- Four bus cycles of Data Bytes (64 bytes)
- One bus cycle of Side band (16 bytes)
- Data and Side Band have different destinations.

6.3 PKTDMA

Multicore Navigator is a methodology and a series of hardware accelerator modules that allow DSP cores and peripherals to effectively transfer packets. It is a safe and managed way that memory can be used to pass data. The Multicore Navigator hardware accelerators decouple DSP cores from the actual transfers elevating what would otherwise be a heavy interrupt and MIPS burden from the DSP core.

Multicore Navigator hardware accelerators use the VBUS infrastructure to implement DMA movement and Queue maintenance. The PKTDMA connection to VBUS is a VBUSM master port and for AIF2, the PKTDMA connection is 128 bits wide.

Figure 6-6. AIF2 Multicore Navigator



The DSP host allocates blocks of memory and configures the Multicore Navigator hardware to utilize the memory region. A key aspect of the Multicore Navigator solution is that the memory is broken up into small grain buffers that are linked listed together via the Multicore Navigator hardware creating virtually any size packets. This approach utilizes the memory space very efficiently minimizing “white space” overhead. While the processing burdened of this linked list approach is significant, much of it is performed in hardware accelerators minimizing any impact on DSP MIPS or peripherals.

There is a trade off when choosing the Buffer size. Small buffers give finer grain usage of memory, but may require more buffers to be linked together to store a packet (creating more overhead in storing/retrieving packets). Larger Buffer size would require few buffers, but the last buffer in a linked list is likely to be only partially full (wasting memory).

Multicore Navigator has a large degree of flexibility. Some applications are not tolerant to breaking up packets into multiple buffers. For these applications, Multicore Navigator has an alternate operation where only a single fixed sized buffer is used per packet. Clearly the buffer size needs to be chosen in advance to handle the largest possible packet size (Each memory region supports only one buffer size).

Figure 6-7. Multicore Navigator Host Mode Example

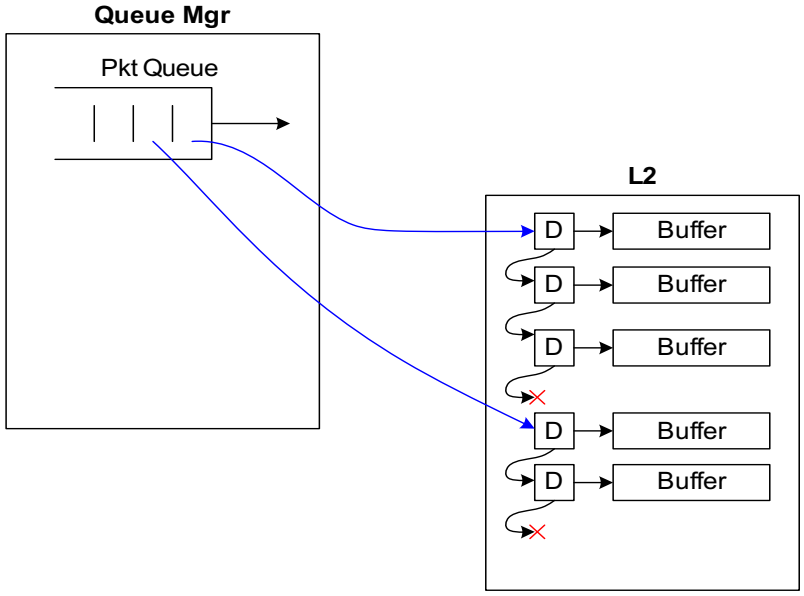
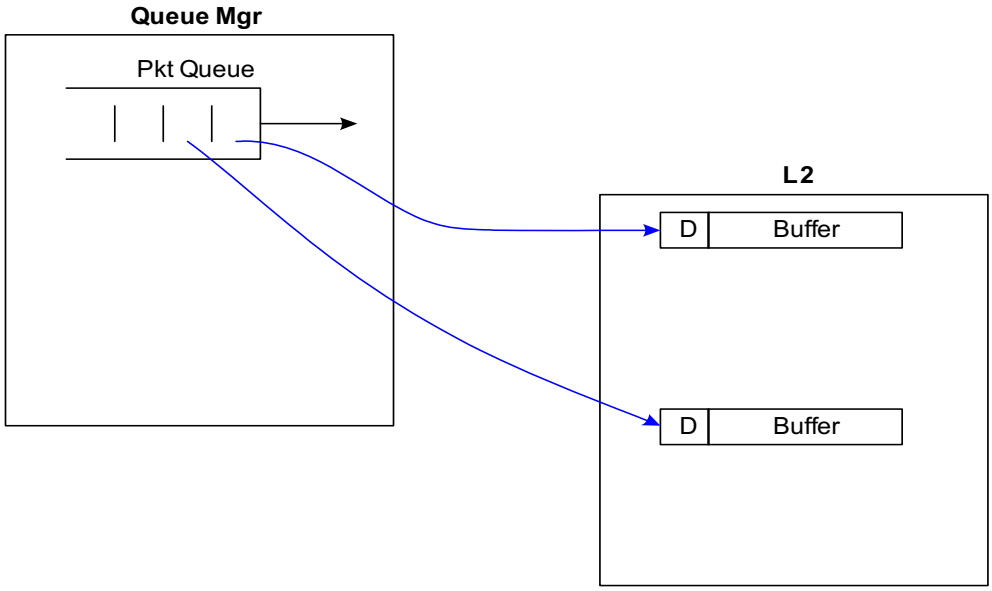


Figure 6-8. Multicore Navigator Monolithic Mode Example



6.3.1 Multicore Navigator Buffers and Descriptors

Multicore Navigator has the concept of Descriptors and Buffers:

- **Buffers**: Buffers hold raw data. The CorePac core allocates a region of memory for buffers and the CorePac core dictates the fixed size for all buffers in this memory region. Multicore Navigator can support multiple memory regions, but each region can support only one buffer size. (Monolithic mode will use exactly one of these buffers per packet. Host mode can used one or more buffers chained together in a linked list)
- **Descriptors**: Descriptors are a form of packet header –or- buffer header which contains information specific to Multicore Navigator. Pointers to Buffers (differing from linked list) are contained in Descriptors as well as many other useful fields both for software and the PKTDMA engine. Descriptor fields vary depending on the Multicore Navigator packet type being supported.
  - **Host Mode Descriptor:** As well as other information, hold an additional pointer the “next” Host

Mode Descriptor in the packet (forming a linked list). The Descriptor and Buffer are separate entities stored in separate areas of memory.

– **Monolithic Mode Descriptor:** Descriptor and Buffer are merged into one contiguous portion of memory (basically, the Buffer contains the Descriptor in the first n words). PKTDMA is aware of the offset between descriptor and beginning of payload.

NOTE: **AxC & Packet Tx Buffers Must be in same memory region.** All traffic for a given AIF2 egress DMA channel must be sourced from the same memory region (i.e. the same L2). The issue which is being avoided is re-ordering of VBUS read requests due to different memory targets. Assume that there are back-to-back memory requests for a single AxC where the first memory read is for a highly loaded L2 and the 2nd memory read is for a lightly loaded L2. PktDMA is a high speed interface and will pipeline issue these read requests prior to the first read request receive data. If the latency of the first read is higher than the pipelined issue and round trip return of the 2nd read request, the data will be presented out of order to the AIF2. This would corrupt the traffic for that AxC possibly scrambling the order of up to 32 consecutive samples. This is not an issue for consecutive requests to the same memory region as all consecutive requests will be returned in order even if read requests experience stalls.

6.3.2 Multicore Navigator Queues

Queues are at the heart of the Multicore Navigator concept. There are several types of Multicore Navigator Queues.

- **Free Queue**: At the time of initialization, all Descriptor/Buffer pairs are pushed onto a Free Queue. Each memory region should be given its own Free Queue. As Descriptor/Buffer pairs are allocated, they are popped off the Free Queue. When Descriptor/Buffer pairs are no longer needed, they are deallocated by Popping off the appropriate queue and pushing them onto the Free Queue.
- **Receive Queues**: Receive and Transmit Queues are packet queues that are used for scheduling PKTDMA transfers. Multiple Queues are useful to implement a Priority scheme. When host receive interrupt from PKTDMA that packet has arrived and populated Receive Queue, host will pop packets from receive queue for processing and the receive queue could be popped into Free Queue for recycling.(this should be done by the host. PKTDMA will not do this automatically)
- **Transmit Queues**: The host Pushes Packets on a Transmit Queue for DMA and PKTDMA will automatically recycle the queue when the packet is fully transferred.

When App software shuts down a channel, there could be residual packet data in the Multicore Navigator buffers for that channel. PD will stop generating Multicore Navigator packet data once the AxC is disabled, but if the App software stops reading this Multicore Navigator traffic, it would be the responsibility of the software to move all the descriptor pointers to a free queue. In the egress Direct IO, after disabling a channel in PE, the PE will cycle through any residual data in the Multicore Navigator channel and “drop it on the floor”.

6.3.3 Multicore Navigator and DIO Scheduling

The Arbitration scheme allows for programmable priority between the following:

- Multicore Navigator Traffic
- Direct IO Traffic

When only one form of traffic has pending data, that traffic is serviced. If both forms of traffic have pending data, the state of the MMR bit indicates which one will be serviced. The higher priority traffic is serviced exclusively until all pending data is transported, then and only then is the other form of traffic serviced.

For the two different modes of DMA, the scheduling is performed in a very different way:

- Direct IO: An internal system event from AT triggers the DIO FSM to burst transfer all DMA activity programmed into the engine. Data is DMA’ed from circular buffers and is transferred without feedback from the PD receive engine.
- Multicore Navigator: Data transfer is scheduled based on the PHY rate of data reception/ transmission

6.3.4 Multicore Navigator Packet Types

Multicore Navigator is extremely general and flexible; the intended use of AIF2 is to use Multicore Navigator in a very specific and limited way of using only two Multicore Navigator packet types:

- Monolithic: Used for {LTE, WiMax, TD-SCDMA, GSM} Antenna data and Control data
- Host: Used for {Control data, Generic packet, Ethernet} traffic that requires a large data buffer

NOTE: It is possible to use Host Mode for antenna traffic instead of Monolithic. If this is desired, please be aware the Host Mode has more descriptor fetch overhead than Monolithic. Make sure that system performance will not suffer.

NOTE: The Monolithic Descriptor type has a variable length. AIF2 uses only a 12 byte descriptor (with an optional four bytes of protocol specific) in order to keep the Descriptor VBUS overhead to one bus cycle.

The Host mode packets are the most general and widely used packet (in other IP peripherals) as it handles variable packet length by chaining *n* buffers together in a linked list fashion. Clearly, if the packet does not completely fill the last buffer, the unused area is wasted. Host mode descriptors are separate from the buffers where each descriptor contains a pointer to a data buffer and a second pointer to the next descriptor in the linked list.

Host mode has a lot of performance overhead with all the descriptor and pointer operations. In Host mode, the first descriptor is known as the packet descriptor and only this descriptor is placed on a transport (DMA) queue in the QM.

During configuration, the Host is responsible to prelink Host Mode descriptors and buffers by writing a buffer pointer to each descriptor. After this operation, all descriptors are placed on a free queue; Once on a free queue, Multicore Navigator can randomly assign buffers as needed, performing memory management.

At the time of configuration, it is the responsibility of the user to program how much memory to allocate and how large the buffers should be. If the buffers are too large, then the memory is not used efficiently. If the buffers are too small, there is a higher performance overhead for increased linking operations.

The Monolithic packet type is mainly envisioned for the purpose of passing OFDM symbol data. It is designed for maximum performance in both clock cycles and memory utilization.

Multicore Navigator Monolithic packet types have the descriptor and buffer concatenated in one contiguous memory buffer. Because the descriptor and buffer are in contiguous memory, the PKTDMA operations are simplified when addressing memory. Monolithic packet types do not support linked list chaining and therefore must be sized correctly to accommodate the largest possible packet.

All traffic for a given AIF2 egress DMA channel must be sourced from the same memory region (that is, the same L2). Assume that there are back-to-back memory requests for a single AxC where the first memory read is for a highly loaded L2 and the 2nd memory read is for a lightly loaded L2. PKTDMA is a high speed interface and will pipeline issue these read requests prior to the first read request receive data. If the latency of the first read is higher than the pipelined issue and round trip return of the 2nd read request, the data will be presented out of order to the AIF2. This would corrupt the traffic for that AxC possibly scrambling the order of up to 32 consecutive samples.

The Monolithic packet descriptor header for AIF2 is exactly 16 bytes (one VBUS 128 data phase). Many of the fields are required by Multicore Navigator, some of the remaining bits (4 bytes) are allocated for protocol specific use. The Monolithic packet type uses some of the protocol specific bits for Radio Standard specific information:

- Ingress/Egress selection
- AxC number
- Symbol Number (or GSM timeslot number)

In the special case of GSM Base Band Hopping where App software is assigning OBSAI address to Control Packets, the protocol specific word contains:

- OBSAI address (13 bits)

Packets that are sourced by the CorePac core are expected to have the Multicore Navigator descriptors properly populated. AIF2 requires CorePac to fill in {Ingress/Egress, AxC num, Symbol num} fields for transmit AxC packets; AIF2 will check and fail packets that do not match expected timing. If these AxC num and Symbol num is not matched with expected timing, PE or PD link will be halted until the end of the Radio frame.

FFTc or any other operations after CorePac core that creates packets is expected to maintain these AIF2 (protocol specific) fields of information as AIF2 requires this metadata. PKTDMA Tx configuration register has special bit field called AIF_MONO_MODE. This bit should be set to one if user wants to use 4 bytes protocol specific field.

NOTE: It is possible for the FFTC to send output packets directly to the AIF2 without any DSP intervention. In that case, the AIF_MONO_MODE should be turned off and FFTC uses a descriptor size field of 16 bytes as normal mode, because FFTC can not set 64 bytes to DESC_SIZE field before pushing the descriptor into the AIF2 TX queue.

PKTDMA loopback mode also should be disabled because the default value is loopback enable. PKTDMA emulation control register in Global control register region has this bit field.

In the event that Monolithic is used for non-AxC data, AIF2 will fill Egress protocol specific fields with zeros and Ingress protocol specific fields will be disregarded.

At the time of Configuration, the user should choose the smallest Monolithic packet to fit OFDM Symbol or GSM timeslot. Monolithic packets can be allocated in *n* x 16bytes. Choose the size to be the symbol size plus 16 bytes for descriptor header (rounded up to the nearest 16 bytes). (for LTE, the first, largest symbol size should be chosen).

The following tables show the example of monolithic descriptor field

Table 6-1. Monolithic Packet Descriptor Word 0

Bits	Name	Description
31-30	2'b10	Monolithic Packet Descriptor type.
29-25	Packet Type	This field indicates the type of this packet and is encoded as follows: 0-31 = To Be Assigned by User
24-16	Data Offset	This field indicates the byte offset from byte 0 of this descriptor to the location where the valid data begins. On Rx, this value is set equal to the value for the SOP offset given in the Rx DMA channel's monolithic control register. When a monolithic packet is processed, this value may be modified in order to add or remove bytes to or from the beginning of the packet. The value for this field can range from 0-511 bytes, which means that the maximum NULL region can be 511-12 bytes, because byte 0 is the start of the 12 byte packet info area. Note that the value of this field must always be greater than or equal to 4 times the value given in the Protocol Specific Valid Word Count field. Set to 0x10 for AIF2 (16 bytes)
15-0	Packet Length	The length of the packet in bytes. The valid range is from 0 to 65536 bytes.

Table 6-2. Monolithic Packet Descriptor Word 1

Bits	Name	Description
31-24	Source Tag - Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_hi_sel field in the flow configuration table entry.
23-16	Source Tag - Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_lo_sel field in the flow configuration table entry. Normally, this field used as an index of Rx flow
15-8	Dest Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_hi_sel field in the flow configuration table entry. Not used for AIF2
7-0	Dest Tag - Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_lo_sel field in the flow configuration table entry. Not used for AIF2

Table 6-3. Monolithic Packet Descriptor Word 2

Bits	Name	Description
31	Extended Packet Info Block Present	This field indicates the presence of the Extended Packet Info Block in the descriptor. • 0 = EPIB is not present (set zero for AIF2) • 1 = 16 byte EPIB is present
30	RESERVED	
29-24	Protocol Specific Valid Word Count	This field indicates the valid # of 32-bit words in the protocol specific region. For AIF2, this should be 1. This is encoded in increments of 4 bytes as follows: • 0 = 0 bytes • 1 = 4 bytes • … • 16 = 64 bytes • ... • 32 = 128 bytes • 33-63 = RESERVED
23-20	Error Flags	This field contains error flags that can be assigned based on the packet type. Not used for AIF2
19-16	Protocol Specific Flags	This field contains protocol specific flags / information that can be assigned based on the packet type. Not used for AIF2
15	RESERVED	
14	Return Push Policy	This field indicates how a Transmit DMA should return the descriptor pointers to the free queues. This field is encoded as follows: • 0 = Descriptor must be returned to tail of queue • 1 = Descriptor must be returned to head of queue This bit is only used when the Return Policy bit is set to 1.
13-12	Packet Return Queue Mgr #	This field indicates which of the 4 potential queue managers in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and should be initialized by the host.
11-0	Packet Return Queue #	This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete.

Table 6-4. Monolithic Packet Descriptor Word 3 (Protocol Specific Info)

Bits	Name	Description
31-16	Reserved	
15	Ingress/Egress	0: Ingress 1: Egress
14-7	Symbol Number	Symbol number (0x00 – 0xFF)
6-0	AxC Number	AxC number (0x00 – 0x7F)

Table 6-5. Monolithic Packet Descriptor Word 3 (Protocol Specific Info Alternate Mode)

Bits	Name	Description
31-16	Reserved	
15	Ingress/Egress	1: Egress Only
14-13	Reserved	
12-0	OBSAI address	13 bit OBSAI address that will be inserted in OBSAI header for GSM frequency hopping

6.4 Multicore Navigator Examples

QM: Multicore Navigator Queue Manager

Packet DMA: PKTDMA Controller

6.4.1 Ingress (AIF2 Reception DMA-to-L2 RAM)

- CorePac Initializes Multicore Navigator (PKTDMA, QM)
  - Buffer Allocated
    - The CorePac core allocates a portion of L2 memory for data sharing between the AIF2 and CorePac.
    - The CorePac programs Multicore Navigator hardware accelerators with the location and size of these buffers.
  - Monolithic and Host Descriptors filled out (Initialized)
    - CorePac core writes a link-pointer in each descriptor pointing to it's associated buffer
  - Queue Manager notified of all available resources
    - Buffer region, number of buffers and buffer size are programmed into queue manager
  - Free Descriptor Queue filled with all Descriptors
    - Free Queue is a virtual queue inside queue mgr (shared)
    - Free Queue is implemented with Linking RAM as are all other Queues
    - CorePac writes all descriptors addresses to Free Queue essentially "pushing" them onto the queue
  - Enable PKTDMA Rx, Tx channels and disable loopback mode
    - Enable each Rx or Tx channels with AIF_MONO_MODE option in case of OFDM.
    - Disable PKTDMA loopback mode.
- **AIF2 starts receiving** packet over OBSAI or CPRI interface
  - AIF2 places packet information in AIF2 shallow buffer in DB
    - Shallow buffer is internal AIF2 FIFO
  - Multicore Navigator/AIF2 DMA Scheduler (AD)
    - Once a programmable number of bytes are in the AIF2 buffer, AIF2 DMA Scheduler passes a "credit" to PKTDMA controller.
    - Each DMA Channel has a default Rx Queue that will be the next destination of the packet.
    - AIF2 Scheduler has an internal queue of "credits" indicating a requested DMA transfer, the channel number, and number of bytes to be transferred.

- AIF2 has several categories of packets that are managed. Each category has its own FIFO and is considered a Multicore Navigator channel.
- AIF2 will usually request 64 byte transfers.

– PKTDMA Control
- If SOP, PKTDMA read (fetch from QM) descriptor over VBUS
- (QM) allocates a descriptor for the new packet
  - This is considered a “packet descriptor” (being the first descriptor per packet)
  - For large packets, multiple descriptors are allocated. Subsequent descriptors are simply called descriptors and also requested by Packet DMA (PKTDMA)
- (PKTDMA) writes to Descriptor (in L2). Fills in Descriptor with information about Packet
- (PKTDMA) Reads pointer to Buffer (from Descriptor in L2)
- (PKTDMA) Writes data to data Buffer

– PKTDMA transfers data
- DMA engine has state RAM for mid-point address
  - MOP is faster than SOP or EOP because of no buffer fetch…
- DMA reads data across Multicore Navigator FIFO I/F (similar to VBUS)
  - Data
  - Info words:
  - SOP, EOP
  - Data_byte_cnt (AIF2 could tie off to that is, 64)
  - Port#, Chan# (all tied off to zero, would go into descriptor)
- If packet is larger than a single buffer, QM, BM, and PKTDMA work together to
  - Assign a new descriptor
  - Link list, link the new descriptor to the old descriptor
  - Allocate a new buffer
  - Link the new descriptor to the new buffer
  - DMA the partial packet FIFO data to the new buffer
- If EOP, Write descriptor contents
  - (PKTDMA) writes Descriptor to Packet Queue
  - (QM) create system event

- CorePac
  – CorePac either polls or waits for interrupt from system event
  – CorePac reads “Packet Queue”
  – Pointer to descriptor
  – Either reads Descriptor, or for monolithic skips header and uses data
  – Host Mode… CorePac will follow links in Descriptors…
  – When CorePac is done with packet
    - CorePac navigates Packet Linked List
    - Writes Descriptors to Free Queue for recycle.

6.4.2 Egress (L2 RAM DMA-to-AIF2 Transmission)

This section is abbreviated. Please see Ingress example for greater detail.

- CorePac Initializes Multicore Navigator
- CorePac packet create
  - CorePac Pops descriptors from Free Queue
  - CorePac Fills in descriptors and buffers, popping and filling new descriptors as needed.
  - CorePac fills in first SOP descriptor and pushes onto Transport Queue
- AIF2 DMA Scheduler controls PKTDMA to transfer packet
  - QM gives AIF2 a constant status of FIFO not_empty for each of 128x FIFOs
  - Scheduler monitors AIF2 output buffer depth, if space available
  - Scheduler issues credits in 64 byte increments, indicating
    - {Chan_Num, Byte_Len}
  - If EOP, Scheduler indicates EOP and number of bytes in last transfer
- PKTDMA performs transfer
  - PKTDMA fetches Descriptor address from QM
  - PKTDMA fetches Descriptor from L2
  - PKTDMA (uses Buffer Address from Descriptor) fetches Buffer from L2
  - PKTDMA writes Buffer info to AIF2 input buffer
  - As Buffer boundaries are encountered (for Host Mode)
    - PKTDMA will push freed descriptors/buffers onto Free Queue for recycle
    - PKTDMA use link in descriptor to fetch next descriptor in linked list
  - As EOP is encountered PKTDMA will close out packet
    - PKTDMA updates Descriptor with {Pkt_Len, Bad_Pkt}
    - DC writes Descriptor pointer to Rx_Queue

Implementation Details and Application for Internal Modules

Topic **Page**

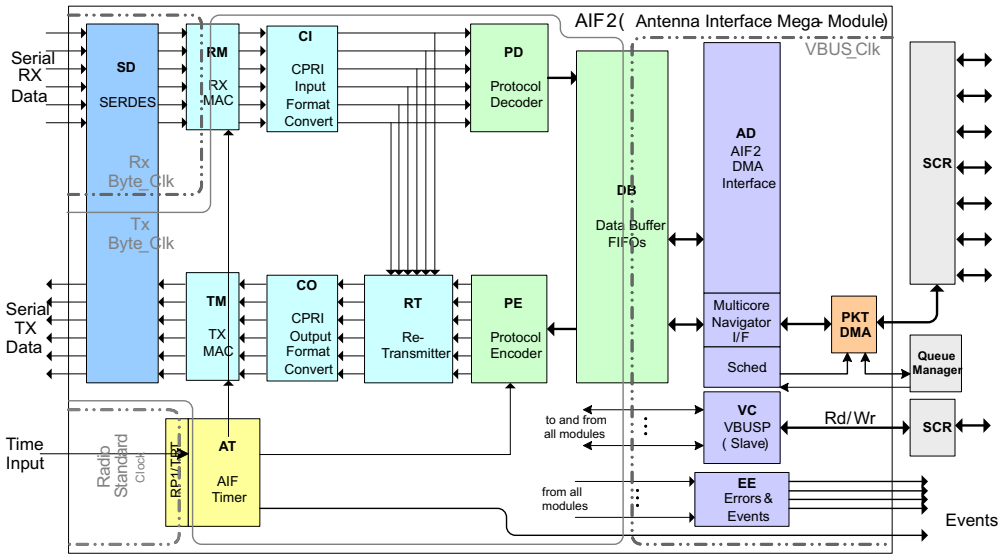
7.1 AIF2 Architecture 122
7.2 VC (VBUSP Configuration Bus Interface) 123
7.3 SD (SerDes) 125
7.4 RM (RX Mac) 131
7.5 TM (TX Mac) 139
7.6 CI and CO (CPRI Input and Output Data Module) 144
7.7 RT (Retransmitter) 150
7.8 PD (Protocol Decoder) 151
7.9 PE (Protocol Encoder) 162
7.10 DB (Data Buffer) 172
7.11 AD (AIF2 DMA) 182
7.12 AT (AIF2 Timer) 190
7.13 EE (Error and Exception Handler) 208

7.1 AIF2 Architecture

Figure 7-1 shows the total AIF2 internal hardware architecture. AIF2 has three basic layers, as follows:

- PHY layer {SD, RM, CI, RT, CO, TM}
- Protocol layer {PD, PE, DB}
- DMA layer {AD, PKTDMA}

Figure 7-1. AIF2 Hardware Architecture



AIF2 is works on a link-by-link basis, so users can easily enable or disable links in each submodule.

The following analysis shows that the DB (Data Buffer) has sufficient bandwidth to handle the peak usage case. DB is TDM shared between six links plus one link of data trace. For both OBSAI and CPRI, there are payload and special control byte bandwidth. For the Data trace feature, the bandwidth equals the byte rate of the link.

While the AIF2 is not required to handle the max bandwidth of all six links in 8x rate mode with all AxC populated, there is a significant chance that AxC data is not spread out evenly within a link, thereby causing a peak bandwidth that would equal to the worst case scenario of all links fully populated (running at 8x rate).

Payload Peak Bandwidth → **2949.12** MByte/sec = 3.84MHz x 4bytes x 32AxC x six Links

Special Control carriers

CPRI CW Peak Bandwidth → **184.32** MByte/sec = 1/16 x 2949.12

OBSAI Ctrl Msg Peak Bandwidth → **147.456** MByte/sec = 1/20 x 2949.12

Data Trace Peak Bandwidth

CPRI ‡ 614.25 MByte/sec = 491.4 * 1.25

OBSAI ‡ 768.0 MByte/sec = 614.4 * 1.25

Total Peak Ingress/Egress Bandwidth

CPRI ‡ 3747.69 MByte/sec = 2949.12 + 184.32 + 614.25

OBSAI ‡ 3864.576 MByte/sec = 2949.12 + 147.456 + 768.0

Max allowable DB Peak Ingress/Egress Bandwidth

CPRI ‡ 3931.2 MByte/sec = 16 x 245.7MHz

OBSAI ‡ 4915.2 MByte/sec = 16 x 307.2MHz

In this chapter, we'll going to look at each module and see how it works and how to configure MMRs and some example configuration will be shown for some cases.

7.2 VC (VBUSP Configuration Bus Interface)

7.2.1 Clock Stop and Emulation

Emulation and clock stop are controlled via higher layer hardware and software functions. AIF2 treats these functions identically. AIF2 performs a controlled and graceful shutdown of local insertion and extraction of traffic AIF2 maintains the Daisy chain as well as possible, transitioning to retransmission operation when a mid-point, or inserting empty frames when an end point.

The controlled shutdown of traffic involves the AIF2 ending extraction and insertion on radio frame boundaries for AxC traffic and ending packet traffic on EOP (end of packet) boundaries.

On recovery from either Clock Stop or Emulation suspend, AIF2 continue to halt extraction/insertion until SW comes in and set both PE & PD. This is intended to allow SW a clean and orchestrated power up sequence.

7.2.2 Emulation Control

The emulation control input (**cdma_emususp**) and register bits (**soft** and **free** in the Emulation Control register) allow DMA operation to be suspended. When the emulation suspend state is entered, the DMA will stop processing receive and transmit packets for each channel at the next packet boundary. Any packet currently in reception or transmission will be completed normally without suspension. Emulation control is implemented for compatibility with other peripherals.

Table 7-1 shows the operations of the emulation control input and register bits:

Table 7-1. Emulation Suspend

emususp	soft	free	Description
0	X	X	Normal Operation
1	0	0	Normal Operation
1	1	0	Emulation Suspend
1	X	1	Normal Operation

rt_sel: RT Emulation mode selection bit

There are two emulation signals that cause the AIF2 to stop, aif2_emu_dbgsusp and aif2_emu_dbgsusp_rt. The rt_sel just selects which one is looked at. If this bit is zero, AIF2 emulation mode is controlled only by the aif2_emu_dbgsusp signal and the aif2_emu_dbgsusp_rt signal is ignored. If this bit is set to one, AIF2 emulation mode is controlled only by the aif2_emu_dbgsusp_rt signal and the aif2_emu_dbgsusp signal is ignored

7.2.3 AIF2 Reset

AIF2 has many clock domains. In the PHY (& Protocol) layer there are seven clock domains that may not be operational when a hardware reset is initiated. AIF2 has a glitch-less clock mux that mux in the vbus_clk in cases where a global (PHY and protocol) reset is initiated or when the SerDes clock is not present.

CorePac cores can be reset while AIF2 continues to run. To implement this properly, software should shut-down DMA, reception and transmission (setting the AIF2 into bypass) prior to the reset of CorePac. Reset Isolation is implemented with two separate reset signals coming from the device core.

A single MMR contains a bit that is used as a software-controlled hardware reset of the AIF2. This bit is OR'ed with the hardware reset pins coming from the device core. With exception of the VBUS infrastructure, the entire AIF2 hardware is reset when the software reset pin is activated.

VBUS infrastructure is exempt from software reset, as resetting this logic could result in crashing the VBUS.

Circuitry that is not reset during software reset includes the following:

- Config VBUS
  - VBUSP Interface
  - internal SCR circuits.
  - vbus_clk to sys_clk retiming bridge
- Data VBUS
  - VBUSM
- Software reset MMR
  - This is not reset otherwise, it would effectively "self-clear" as a function of software reset

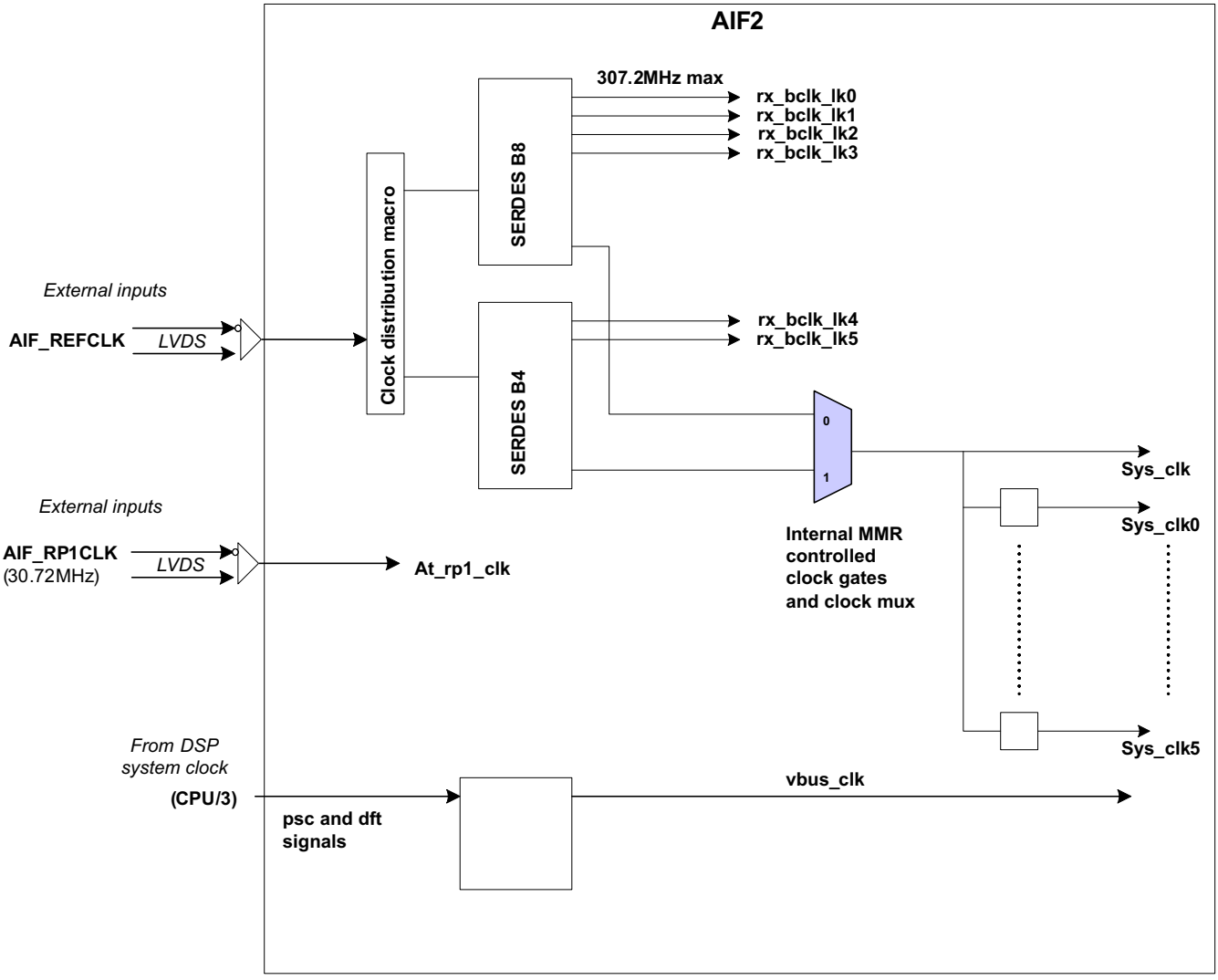
AIF2 CSL supports an API called AIF2_reset(); this API activates the software reset process include MMR flushing.

7.3 SD (SerDes)

7.3.1 AIF2 Clock Strategy

Clocking of the AIF2 processing occurs in the TX byte clock domain that is typically 307.2MHz (OBSAI) or 245.76MHz (CPRI). Interface to the system will be in the CPU clock/3 clock domain that we call vbus_clk internally. Figure 7-2 shows AIF2 clocking sources and how clocking is connected internally.

Figure 7-2. AIF2 Clocking



AIF2 has two special SD common register. One is SD Clock select configuration register and the other one is SD Clock disable configuration register. User can choose the source of Tx byte clock from either B8 or B4 SerDes macro by using clock select configuration register. Clock disable configuration register is used to save power by closing gate off of the unused link.

This UG is using several expressions for AIF2 operation clock. Basically, we use Tx dual byte clock, byte clock, sys clock, 307.2 MHz clock (OBSAI and 5x CPRI), 245.76 MHz clock (CPRI) as a same concept. Only RP1 clock (30.72 MHz) and VBUS clock (CPU clock/3) shows different concept.

7.3.2 SerDes Serial Bit Ordering

The serial coding provides the 20-bit encoded symbol to the SerDes module such that it follows the bit order defined for both the OBSAI RP3 and CPRI standards. These bit orderings for each 10 bits of the 20-bit pair are shown in the following figures. The SerDes module expects that the first bit transmitted or received is bit 0 of the Parallel interface of the module.

Figure 7-3. OBSAI Serial Bit Ordering

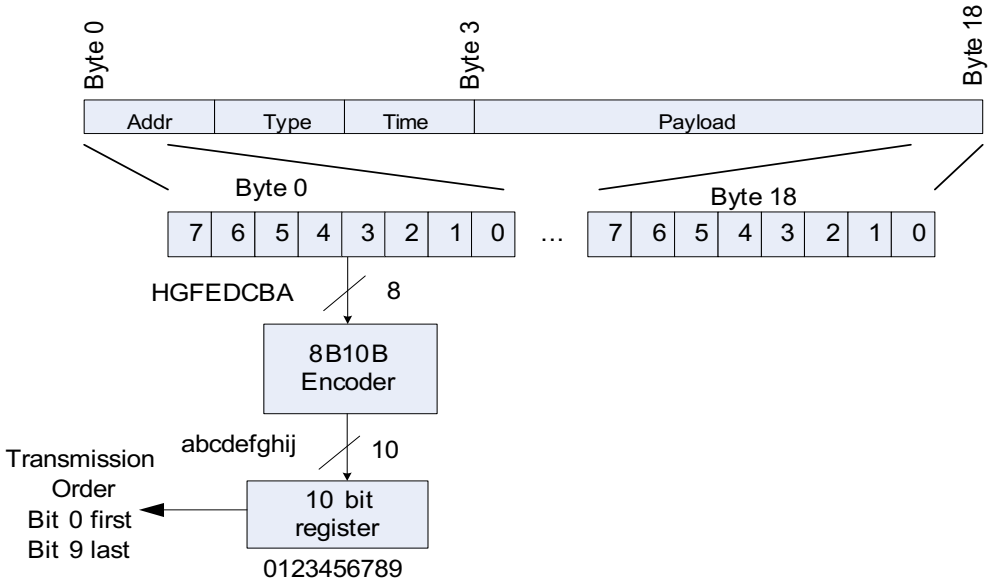
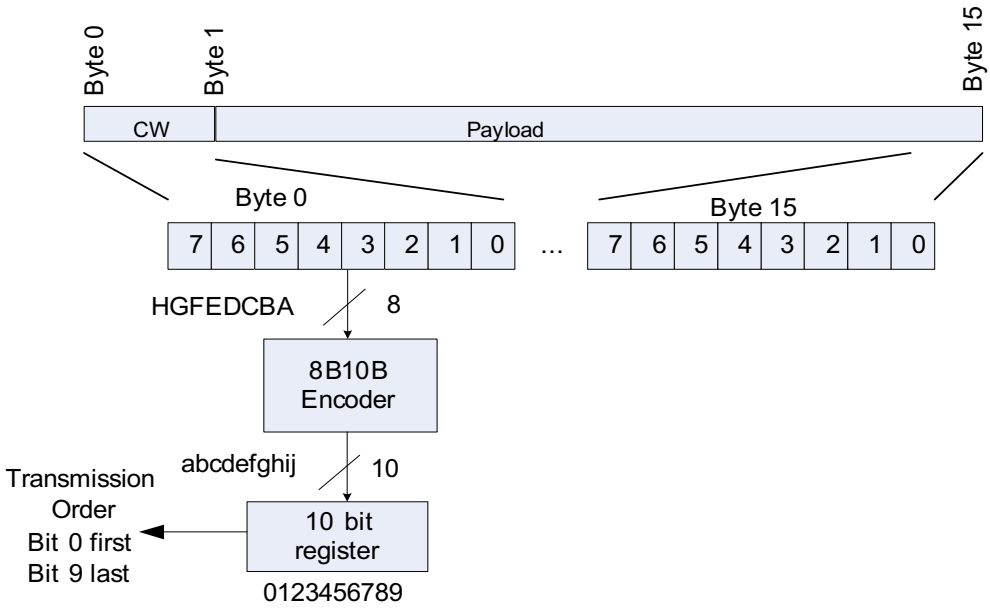


Figure 7-4. CPRI Serial Bit Ordering



7.3.3 SerDes Common Configuration

SD module has two Macros (B8, B4) and each Macro has its own PLL circuit inside and users can enable or disable each PLL by setting **SD_PLL_B8_EN_CFG** or **SD_PLL_B4_EN_CFG** register and check each PLL status using **SD_PLL_B8_STS** or **SD_PLL_B4_STS** register.

The PLL can be put into a low power sleep state by asserting **SLEEP PLL** high. In this mode, the core of PLL remains locked to **refclkp/n**, but all clock distribution and associated circuits are powered down.

7.3.3.1 MPY Setup

During normal operation the integrated PLL uses refclkp/n to generate a higher frequency clock from which the bit rate can be derived. refclkp/n can be in the range 100 - 800MHz nominal. The clock generated by the PLL will be between four and 25 times the frequency of refclkp/n, according to the multiply factor selected via the **MPY** field.

For example, if refclkp/n is 384 MHz, the MPY should be 8x to make 307.2 MHz dual byte clock (handle two 10 bit data chunks per clock). AIF2 SerDes supports 12 GHz serial line rate for full rate, 6GHz for half rate, 3GHz for quarter rate and 1.5 GHz for eighth rate. Full rate is not used to support real link speed. Half rate is used for 8x link rate, Quarter rate is used for 4x link rate and Eighth rate is used for 2x link rate. Quarter rate is also used for 5x link speed for CPRI but internal clock logic is more complex than normal 4x case.

The PLL VCO must also be within the nominal range 1.5625 - 3.125 GHz

7.3.3.2 VCO Speed Range

It is necessary to adjust the loop filter depending on the operating frequency of the VCO. To indicate the selection the user must set the **VRANGE** bit. If the VCO is running at the lower end of frequency range the **VRANGE** should be set high.

If LINERATE × RATESCALE < 2.17GHz, **VRANGE** should be set high.

The relation between Line rate and Rate scale is in Table 7-2 (OBSAI case).

Table 7-2. Line Rate vs. PLL Output Clock Frequency

Link Rate	Line Rate	PLL output frequency	Rate scale
8x (Half)	6.144 Gbps	0.5 * Line Rate Ghz	0.5
4x (Quarter)	3.072 Gbps	Line Rate Ghz	1
2x (Eighth)	1.536 Gbps	2 * Line Rate Ghz	2
5x (Quarter)	3.75 Gbps	1.25 * Line Rate Ghz	1.25

refclkp/nFREQ = (LINERATE × RATESCALE)/ ***MPY***

The wide range of multiply factors combined with the different rate modes means it will often be possible to achieve a given line rate from multiple different **refclkp/n** frequencies. The configuration that utilizes the highest **refclkp/n** frequency achievable is always preferable.

7.3.3.3 refclkp/n Jitter and PLL Loop Bandwidth

Jitter on the reference clock will degrade both the transmit eye and receiver jitter tolerance thereby impairing system performance. A good quality, low jitter reference clock is necessary to achieve compliance with most if not all physical layer standards. To minimize the introduction of additional on-chip jitter the differential reference clock inputs **refclkp/n** must be driven from a low jitter clock input buffer (LJCB) or from an on chip LC-based cleaner PLL.

Table 7-3 summarizes how to select one of the available settings. The setting to use depends on the application and the reference clock frequency. The PLL bandwidth is a function of the reference clock frequency, as shown in the equation in Section 7.3.3.2. The value of BWSCALE is a function of both ***LB*** and the PLL output frequency, as shown in Table 7-3.

PLLBW = (**refclkp/n**FREQ)/BWSCALE

Table 7-3. PLL Loop Bandwidth Selection (MHz)

Value	Effect	BWSCALE vs PLL Frequency		
		3.125 GHz	2.1GHz	1.5625GHz
00	Medium Bandwidth.	13	14	16
01	Ultra High Bandwidth	7	8	8
10	Low Bandwidth	21	23	30
11	High Bandwidth	10	11	14

NOTE: Note that use of the ultra high bandwidth setting (***LB*** = 01) is not recommended for PLL multiply factors of less than 8.

7.3.3.4 Clock Bypass

All macros incorporate the ability to bypass the PLL, primarily for manufacturing test purposes.

It can also be used to operate the macro many times slower than normal speed, as might be required if the link partner is a prototype implemented in an FPGA or hardware emulator for example. For normal case, this should be set to 00, no bypass option.

7.3.4 SerDes Rx Configuration

7.3.4.1 Data Rate

The primary operating frequency of the SerDes macro is determined by the reference clock frequency and PLL multiplication factor. However, to support lower frequency applications, each receiver can also be configured to operate at a half, quarter, or eighth of this rate via the ***RATE*** bits as described below.

- Full rate—(Four data samples taken per PLL output clock cycle) AIF2 does not support this.
- **8x—**Half rate. (Two data sample taken per PLL output clock cycle)
- **4x, 5x—**Quarter rate. (One data sample taken per PLL output clock cycle)
- **2x—**Eighth rate. (One data sample taken every two PLL output clock cycles)

7.3.4.2 Symbol Alignment

Symbol alignment based on K28 comma symbols is utilized by all standards that use 8b10b encoded data. Setting **ALIGN** to 01 will enable alignment of the received data stream based on comma symbols. However, it is normal to disable comma detection by setting ALIGN to 00 once some number of aligned commas have been received.

SYNC indicates that an aligned comma has been received. This prevents inadvertent realignment occurring if a bit error changes another symbol into a comma. Typically, small hardware state machines for each receiver channel is required to control ALIGN. The exact alignment protocol is dependent on the standard being used.

7.3.4.3 Clock Recovery Algorithms

The clock recovery algorithms operate to adjust the clocks used to sample rxpi and rxni, so that the data samples are taken midway between data transitions. Both first and second order algorithms are provided. Settings 000 through 011 have the 2nd order loop enabled, whereas 100 through 111 have it disabled. There are also four different threshold options (1, 3, 7, 15).

Setting 110 being the default in synchronous applications (that is, where the same clock source is used at both end of the link). If the crystals are different, there will be a constant ppm offset between the two ends of the link, and in this case it is best to enable the second order loop (that is, setting 011 as default). Options that use increased voting thresholds may offer improved jitter tolerance performance in some applications, but this is difficult to predict (really need to perform a detailed matlab type analysis, or try it empirically).

Setting 111 should only be used in short reach synchronous chip to chip applications where squeezing out as much power as possible.

7.3.4.4 Rx Termination

This value should be set to 001: **Common point set to 0.7vddt**. This configuration is for AC coupled systems.

The transmitter has no effect on the receiver common mode, which is set to optimize the input sensitivity of the receiver. Common mode termination is via a 50pF capacitor to **vsst**.

7.3.4.5 Equalization, EQ Hold, and ENOC

Fully adaptive mode (setting 001) could be appropriate for most applications and **EQHOLD** should be set to zero.

NOTE: Offset compensation is required to obtain best receiver performance. This is enabled by setting ***ENOC*** high, which should be the default setting. The ability to turn this off should be provided, and can be shared across all similarly configured receivers.

7.3.4.6 Test Pattern and Loopback Setting

Pattern generation, verification and loopback modes are controlled via **TESTPATT** and **LOOPBACK**.

If pattern generation, verification or loopback capabilities are implemented, these inputs need to be configurable. Unless per channel control is required, it is acceptable to control all the channels in parallel. **TESTFAIL** indicates errors during pattern verification, and can be used to set a status flag or increment a counter.

AIF2 does not support Bump pad loopback, which was in AIF1. The LOOPBACK field should be set to 2b11.

7.3.5 SerDes Tx Configuration

7.3.5.1 Data Rate

The primary operating frequency of the SerDes macro is determined by the reference clock frequency and PLL multiplication factor. AIF2 utilizes the system clock of 307.2 MHz generated by one of the transmit links, all the links must be set to 8x to activate any internal module link rate (2x, 4x, 5x, 8x) and at least this field should be set to 0x1 to activate other module before activating TX SERDES. **This field should be set to Half rate(8x) for all six links at initialization time before locking SD pll.**

Full rate. (Four data samples taken per PLL output clock cycle) Aif2 doesn't support this rate

8x: Half rate. (Two data sample taken per PLL output clock cycle)

7.3.5.2 MSYNC

AIF2 has six lanes; data is transmitted across multiple transmit lanes. During transmission, data is not physically synchronous, but aligned using special symbols in the data stream, so it needs to be able to treat all lanes as one logical clock domain. Transmit lanes that have a lower-numbered neighbor of a different rate or width configuration must have set this bit high. Failing to do so prevents the lane from operating correctly.

To achieve this, the following configurations must be made:

1. Lanes being aggregated must have set this bit low.
2. One macro must be nominated as the master.(B8 if it is used)
3. The master macro must have one lane which tied this high. (lower numbered lane)
4. All other lanes on all other macros being aggregated should be tied low.

Figure 7-5. Infra-macro Case

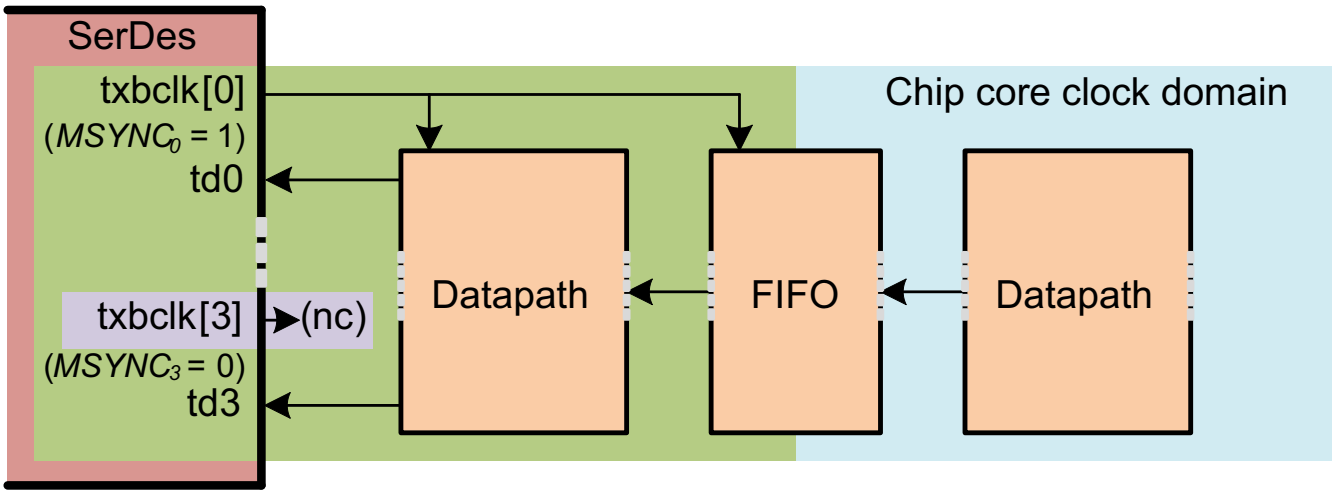
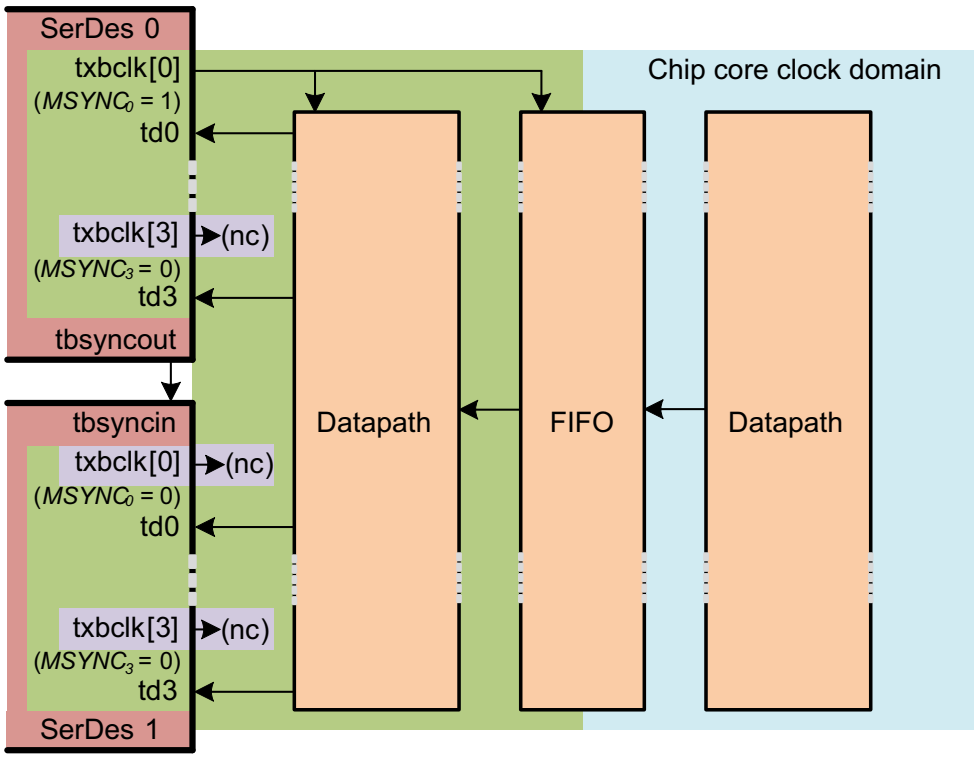


Figure 7-6. Inter-macro Case



For Infra-macro, one txbclk output is used to launch data towards the td bus inputs of multiple Tx lanes on the same macro. For Inter-macro, one txbclk output is used to launch data towards the td bus inputs of multiple Tx lanes of multiple macros. Sync propagates from master to slave macro(s) and the launch clock can be from master or slave.

7.3.5.3 Swing, TWPRE and TWPST

The standard specifies several compliant interconnect and associated far-end eye templates. The swing and equalization settings required depends on which of these templates is being targeted by the application. If the Equalization function is not used, these setting should be 0000 for all.

7.3.5.4 FIRUPT

FIRUPT can be used to ensure changes to equalization settings occur at exactly the same time. If this level of precision is not required, it can be tied high. This is generally the case for data rates below 6 Gbps

7.3.5.5 Test Pattern and Loopback Setting

It is same to Rx configuration and loopback setting is 2b11.

7.4 RM (RX Mac)

The Rx Mac block provides interconnect to the AIF2 IP block from the SerDes macro receive interface of the external SerDes module. The Rx Mac function is implemented as six identical blocks with six separate data paths. Each instantiation of the Rx Mac is provided a separate AT Module Interface for system synchronization.

The Rx Mac (RM) block supports the data path connection from a SerDes module receive link of the AIF2 IP block. The Rx Mac converts a two byte wide parallel stream in the 2x_byte clock domain of the received signal that is clocked directly from the SerDes module. This two byte wide data stream is later converted into a four byte wide stream in the system clock domain. The RM also recovers framing, provides serial link decoding, and performs clock synchronization of either an OBSAI or CPRI signal. The Rx Mac also provides “raw” data to the DB as a data trace feature.

7.4.1 Receive Clock Synchronization

Receive Clock Synchronization is performed by the Rx FIFO, a 32 - dual byte (64 bytes) clock crossing FIFO. This FIFO is written two bytes at a time and is read two bytes at a time. The FIFO also allows for up to ± 52 ns of signal drift (6144 Mbps line rate). The FIFO is written using the receive 2x_byte clock from the associated SerDes macro receiver and is read using the system clock (307.2 MHz or 245.76MHz). The data is read when there is *fifo_thold* amount of data in the FIFO. This water mark value is selectable in a configuration register for either zero 2x_byte clocks (read immediately), four 2x_byte clocks, eight 2x_byte clocks or sixteen 2x_byte clocks. Overflow status is also provided.

This FIFO feeds two destinations. The first is the RM Frame Synchronizer, and the second is the data trace feature in the DB. The data trace feature requires “raw” data (that is, exactly as it exits the 8b10b decoders) that can be analyzed for debugging purposes.

Receive clock monitoring and detection is provided in the clock synchronization block. These circuits interface to the MMR interface of the RM to provide system status.

7.4.1.1 Clock Detection Function

The AIF2 IP block contains dedicated circuitry to detect the validity of a receiver clock signal from the SerDes module. The clock detection is useful in determining AIF2 functionality during failures, and to help isolate causes of a failure in the data path. Clock detection circuitry is provided within each link of the RM block.

The clock detection circuit provides the following features:

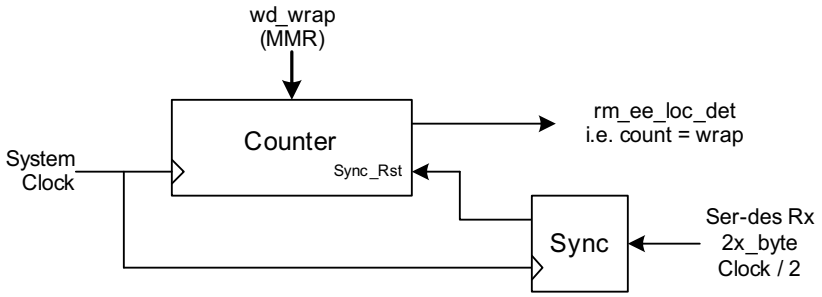
- Clock detection available for each link
- Watch dog indicator
  - Monitors clock activity
  - Generates an event if clock is not active
  - Programmable wrap that determines detection window
- Clock Quality indicator
  - Creates a clock quality value in a status register
  - Programmable wrap register that determines accuracy

The clock detection function is divided into two components. One component provides a watchdog capability designed to indicate weather the Rx 2x_byte clock is not present or not. The second component determines the quality of the clock signal received from the SerDes module.

The Clock Detect Watchdog circuit consists of a programmable counter with a wrap detection that generates an event. The value of the wrap is set through an MMR. If the counter does wrap, an error event (EE) is triggered.

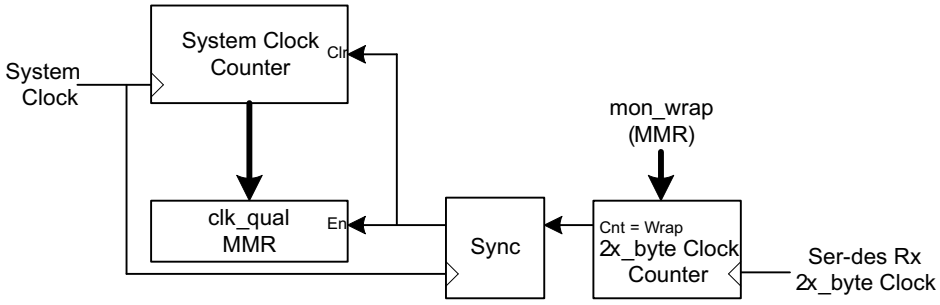
NOTE: The counter is reset *every other* Rx, 2x_byte clock cycle.

Figure 7-7. Clock Detect Watchdog



The clock quality circuit is designed to count the number of received clock edges with the known good quality system clock. Two counters are used, one that counts the number of system clock edges, and another that counts a number of receiver clock edges. Upon a programmable wrap of the receiver clock count, the system clock count is captured. The captured value can be monitored in an MMR. In this way, the Clock Quality counter acks as a frequency counter. For example: system clock = 307.2 MHz, 2x_byte clock = 76.8 MHz (4x link rate), MMR *mon_wrap* = 100. Every 100, 2x_byte clocks the 2x_byte clock counter wraps, clears the system clock counter and loading the system clock count into the *clk_qual* status MMR. During the 100 2x_byte clocks, the system clock counter will have counted approximately 400 system clock periods. The frequency of the 2x_byte clock can then be confirmed to be *mon_wrap* / *clk_qual* * system clock = 100 / 400 * 307.2 MHz = 76.8 MHz.

Figure 7-8. Clock Quality Measurement



Due to clock crossing uncertainties, values for mon_wrap should be kept well under the maximum value specified. Failure to do so may produce erroneous results.

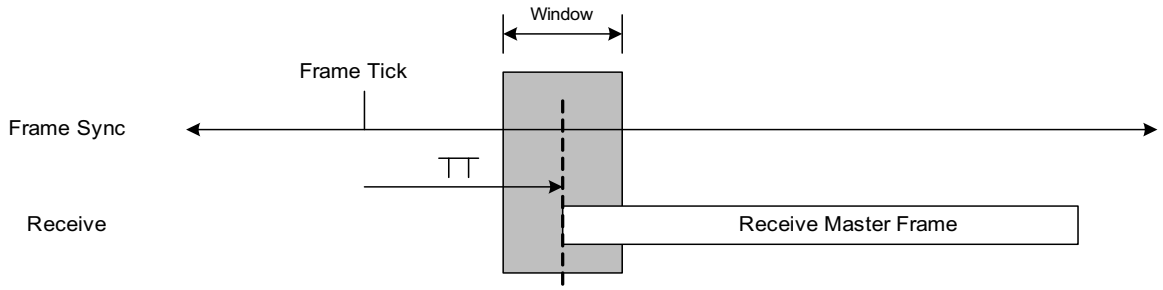
Table 7-4. Maximum allowable mon_wrap Value

Line Rate	mon_wrap value (max)
8x	2^16-1
4x	2^15-1
2x	2^14-1

7.4.2 Receive Frame Synchronization

Receive Frame Synchronization provides a single pulse for the AT module that indicates the beginning of a master frame based on the reception of a K28.7 character and Rx FSM in state ST3 (OBSAI). The pulse represents the Pi variable and is used by the AT module to perform Pi measurement. In CPRI mode, the AT will receive an indication that a K28.5 character occurred and is also qualified with Rx FSM in state ST3.

Figure 7-9. Receive Pi Measurement (In AT Module)



7.4.2.1 Receive Frame Synchronizer State Machine

7.4.2.1.1 OBSAI RP3 Frame Synchronization

The receiver state machine consists of four states for 768, 1536, and 3070 Mbps line rates and six states for 6144 Mbps line rate. The four states applied to all line rates are as follows:

- UNSYNC (ST0)
- WAIT_FOR_K28.7_IDLES (ST1)
- WAIT_FOR_FRAME_SYNC_T (ST2)
- FRAME_SYNC (ST3

Two of these states, WAIT_FOR_K28.7_IDLES and WAIT_FOR_FRAME_SYNC_T, can be considered to form a single logical state called SYNC. The meaning of states UNSYNC, SYNC, and FRAME_SYNC is as follows:

- UNSYNC: Bus link is down. Many byte errors are detected.
- SYNC: Bus link is working, that is, connection exists.
- FRAME_SYNC: Normal operational mode. Frame structure is detected and messages are received.

Two additional states are applied for the 6144 Mbps line rate due to scrambling seed transfer from the transmitter to the receiver: WAIT_FOR_SEED (ST4) and WAIT_FOR_ACK (ST5). These two states form a logical state called SCR_CAP (scrambling seed capture). It is expected, that the receiver can capture scrambling code both from IDLE_REQ and IDLE_ACK patterns. When in the state WAIT_FOR_K28.7_IDLES, IDLE_REQ is detected if every 17th byte of received data is a K28.5 and there are valid data bytes (no K-codes, no LCV errors) between K-codes. Contents of data bytes are not checked. This way it is possible to recognize IDLE_REQ, even if the scrambling code has been changed.

To reduce false recognition of IDLE_REQ, due to an errant K28.5, IDLE_REQ will not be detected in the WAIT_FOR_FRAME_SYNC_T and FRAME_SYNC states. If a scrambling code has been changed, many IDLE_REQs will be received. This will eventually cause FRAME_UNSYNC_T invalid messages groups and force the transition to the WAIT_FOR_K28.7_IDLES state. IDLE_REQ will be detected in this state and cause a transition to the WAIT_FOR_SEED state.

The receiver state machine uses two separate criteria to determine the quality of a bus link. The first one monitors the signal quality by counting LCV errors and the second one monitor the validity of the received frame structure. Parameters BLOCK_SIZE, SYNC_T, UNSYNC_T, FRAME_SYNC_T and FRAME_UNSYNC_T control the transitions.

7.4.2.1.2 CPRI Frame Synchronization

The CPRI Specification illustrates an example of LOF and HFNSYNC detection. The AIF2 goes a step further by making the number of transitions programmable. The same counters (*sync_t*, etc.) that are required for the OBSAI RP3 state machine are used for this purpose.

There are four states defined: XACQ1, XACQn, XSYNC *n* and HFNSYNC. The XACQ1 state is equivalent to the RP3 UNSYNC state. In this state, the bus link is down and byte errors are being detected. This is the state that the Rx Mac comes up in after reset and also when LOS is detected. While in the XACQ1 state, *lof_state* is set true '1'. The synchronization counters are cleared at the transition from state ST0 to ST1.

When the Idle byte K28.5 is detected AND LOS = 0, the XACQn state is entered. If the Idle byte is not detected and the y_cntr, w_cntr and x_cntr sync counters have all counted back to zero, the state machine reverts back to the XACQ1 state. If, however, Idle bytes are detected at the correct time, the state machine will transition to the XSYNC *n* state. The number of correct Idle byte detections before proceeding to the next state is programmable via the *sync_t* counter. When in the XACQ *n* state, *lof_state* is set true '1'.

While in state XSYNC *n*, if an Idle byte is not detected and the y_cntr, w_cntr and x_cntr sync counters have all counted back to zero, the state machine will revert back to the XACQ1 state. The number of incorrect Idle bytes before making this transition is programmable via the *unsync_t* counter. If, however, idle bytes are detected at the correct time, the state machine will transition to the HFNSYNC state. The number of correct Idle byte detections before proceeding to this state is programmable via the *frame_sync_t* counter. When in the XSYNC *n* state, LOF is set false '0'.

The state machine will stay in the HFNSYNC state as long as Idle bytes are detected at the correct time. If, however, Idle bytes are not detected at the correct time, the state machine will transition to the XSYNC *n* state. The number of incorrect Idle byte detections before reverting to the XSYNC *n* state is programmable via the *frame_unsync_t* counter. While in the HFNSYNC state, LOF is set false '0'.

The following comprise the CPRI synchronization counters:

- y_cntr = Byte Number within a word (0 to 1, 0 to 3, 0 to 4 or 0 to 7 depending on link rates 2x, 4x, 5x or 8x respectively)
- w_cntr = Word counter in a Basic Frame (0 to 15)
- b_cntr = Byte counter per word (0 to 63)
- x_cntr = Basic Frame counter (0 to 255)

7.4.2.1.3 OBSAI vs. CPRI Frame Synchronization

Table 7-5 shows Rx sync FSM state transition for OBSAI and CPRI.

Table 7-5. RX Sync FSM State Transition

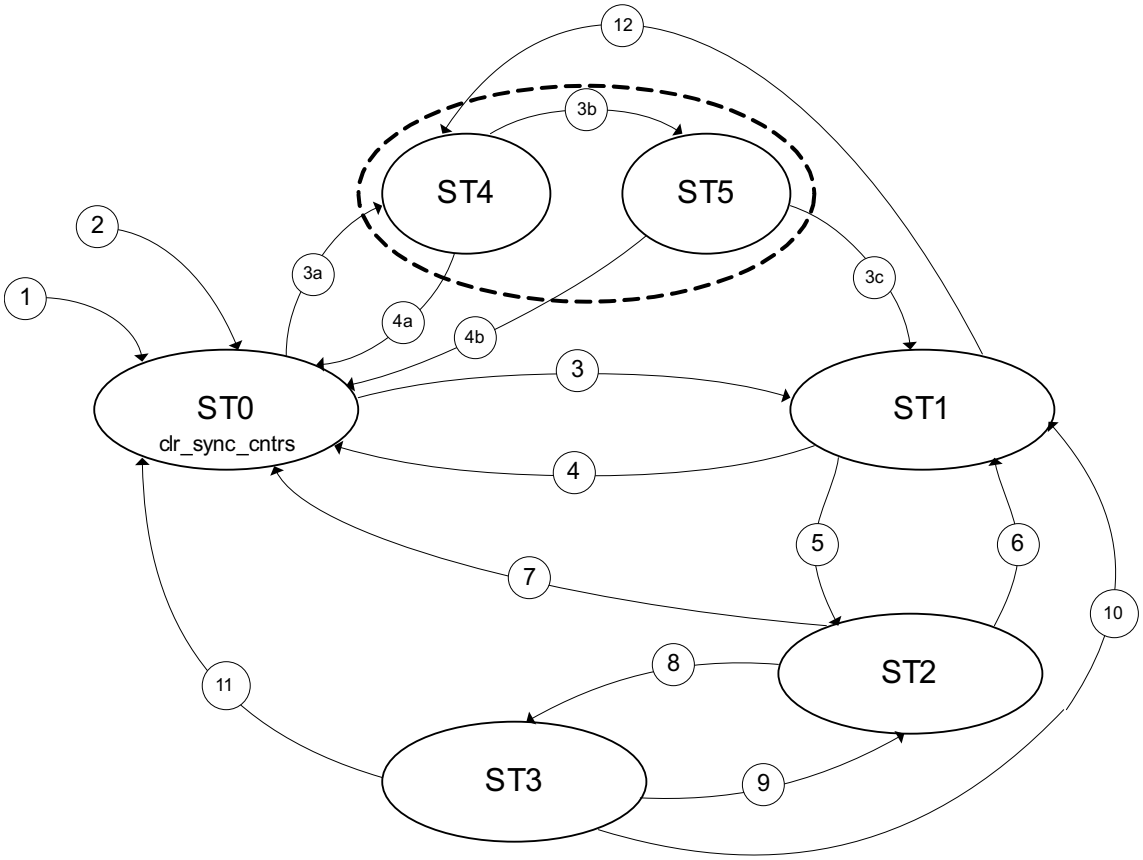
Transition	OBSAI RP3 Description	CPRI Description
1	rst_n	rst_n
2	!rx_en | sd_rm_los_dtct	*!rx_en* | num_los_det (that is, N * 8b10b errors in a Hyperframe) | *sd_rm_los_dtct*
3a	*scr_en* and *sync_t* consecutive valid blocks of bytes received	n/a
3b	*scr_en* and Scrambling seed captured and verified	n/a
3c	*scr_en* and Acknowledge training pattern received	n/a
3	*! scr_en* and *sync_t* consecutive valid blocks of bytes received	K28.5 byte received and ~(num_los_det | *sd_rm_los_dtct*)
4a	*unsync_t* consecutive invalid blocks of bytes received	n/a
4b	*unsync_t* consecutive invalid blocks of bytes received	n/a
4	*unsync_t* consecutive invalid blocks of bytes received	K28.5 byte !received and (y_cntr = 0 and w_cntr = 0 and x_cntr = 0)
5	One K28.7 Idle byte received	(K28.5 byte received and (y_cntr = 0 and w_cntr = 0 and x_cntr = 0)) occurs *sync_t* consecutive times

Table 7-5. RX Sync FSM State Transition (continued)

Transition	OBSAI RP3 Description	CPRI Description
6	*frame_unsync_t* consecutive invalid message groups received	n/a
7	*unsync_t* consecutive invalid blocks of bytes received	(K28.5 byte !received and (y_cntr = 0 and w_cntr = 0 and x_cntr = 0)) occurs *unsync_t* consecutive times
8	*frame_sync_t* consecutive valid message groups received	(K28.5 byte received and (y_cntr = 0 and w_cntr = 0 and x_cntr = 0)) *frame_sync_t* consecutive times
9	n/a	(K28.5 byte !received and (y_cntr = 0 and w_cntr = 0 and x_cntr = 0)) *frame_unsync_t* consecutive times
10	*frame_unsync_t* consecutive invalid message groups received (Idle order matters)	n/a
11	*unsync_t* consecutive invalid blocks of bytes received OR (lcv_unsync_en and num_los_det)	n/a
12	IDEL_REQ pattern detected	n/a

The transition number is in Figure 7-10.

Figure 7-10. Rx Mac Receive State Machine



The receiver synchronization state machine has several outputs. Current state of the receiver (sync_status) is available for the application layer and an event is generated from each state change (sync_status_change). The MMR status bit loss_of_signal is active '1' in state ST0 while it is inactive in other states. For CPRI, the MMR status bit hfnsync_state is also created when in FSM state ST3 and the signal lof_state is created when in either ST0 or ST1.

Table 7-6. Rx Sync FSM Output

State	OBSAI RP3 Output	CPRI Output
ST0	los_state = 1	los_state =1 lof_state = 1
ST1	N/A	lof_state = 1, Set x, y and w counters to 0 at transition from ST0 to ST1.
ST2	N/A	N/A
ST3	frame_sync_state = 1	hfsync_state = 1, frame_sync_state = 1
ST4	N/A	N/A
ST5	N/A	N/A

The receive synchronization state machine states may be forced with the force_rx_state configuration bits. Once in the forced state, the FSM will remain in this state unless forced to another state or the forcing is removed. Other than not being able to change the FSM states when forced, the RM should behave as normal for that state condition.

Table 7-7. Rx Sync FSM State Names

State	OBSAI RP3 Output	CPRI Output
ST0	UNSYNC	XACQ1
ST1	WAIT_FOR_K28.7_IDLES	XACQn
ST2	WAIT_FOR_FRAME_SYNC_T	XSYNCn
ST3	FRAME_SYNC	HFNSYNC
ST4	WAIT_FOR_SEED	N/A
ST5	WAIT_FOR_ACK	N/A

7.4.3 Receive Frame Formatting

Once frame synchronization is established, the Receive Frame Formatting logic provides alignment signals that are “out of band” from the data path. These signals indicate a master frame boundary and message group boundary in OBSAI RP3 mode, and a Basic frame boundary, Hyper-frame boundary and CPRI 10 ms frame boundary for CPRI mode. These alignment signals are used to pipeline the data path to the AIF2 protocol control and DMA logic. Other “out of band” signals associated with the data path are the per-byte K character indication and the 8b10b line code violation error indication.

7.4.3.1 Line Code Violation Handling

For both OBSAI and CPRI, each data byte will have an associated “side band” signal that indicates an 8b10b Line Code Violation.

The special character, K30.7, indicates a received data error. The RM will detect this and create an error event as well as convert the associated payload data byte to 0x00 (OBSAI only)

7.4.3.2 Received K character Handling

The RM will be tolerant of bit errors occurring in K28.5 and K28.7 IDLE bytes. Missing IDLE bytes will be flagged as an error and the RM will insert the missing IDLE byte.

7.4.4 RM Programming

7.4.4.1 RM Link Configuration

The RM Link Configuration Registers are used to program basic functionality of each Rx Mac.

fifo_thold - 2 bits

Sets the watermark of where the RM begins reading from the Rx FIFO

- 00: FIFO starts reading after one dual byte received
- 01: FIFO starts reading after four dual bytes received
- 10: FIFO starts reading after eight dual bytes received
- 10: FIFO starts reading after 16 dual bytes received

error_supress - one bit

Suppress error reporting when the receiver state machine is not in state ST3.

- 0: Allow all RM error reporting when !ST3
- 1: Suppress all RM error reporting when !ST3

sd_auto_align_en - one bit

Enables the RM to automatically disable SerDes symbol alignment when the receiver state machine reaches state ST3. Disabling comma alignment may be necessary in OBSAI mode due to the K28.7 comma character, in combination with certain data, that tricks the SerDes into falsely realigning.

- 0: Disable auto alignment
- 1: Enable auto alignment

scr_en - 1 bit

Enables the scrambler for OBSAI 8x link rate operation in the receive data path.

- 0: De-scrambler Disabled
- 1: De-scrambler Enabled

lcv_unsync_en – one bit

Enables a state transition from the ST3 (FRAME_SYNC) to the ST0 state (UNSYNC) when los_det_thold is met (OBSAI only).

- 0: lcv_det_thold has no effect on the Rx FSM
- 1:The Rx FSM transitions from ST3 to ST0 when lcv_det_thold is met

lcv_cntr_en - one bit

Writing a 1 to this bit will enable the Line Code Violation counter. This 16-bit counter will saturate when it reaches a value of 0xffff. Writing a 0 to this bit will clear and disable the counter. The curent counter value is available as status (*lcv_cntr_value*).

- 0: lcv_cntr disabled and cleared to a value of 0x0000
- 1: lcv_cntr enabled (counts each LCV to a max of 0xffff)

los_det_thold - 16 bits

Sets 8b10b los detect threshold values in number of Line Code Violations received during a master frame (OBSAI) or during a Hyperframe (CPRI). Writing to this location will automatically clear the *num_los* counter and *num_los_det* status bit.

7.4.4.2 RM FSM Sync Count Configuration

Defines the threshold values to reach ST1 and ST3 (FRAME_SYNC) states.

sync_t - 16 bits

Threshold value for consecutive valid blocks of bytes that result in state ST1. Range: 0 to 65,535

frame_sync_t - 16 bits

Threshold value for consecutive valid message groups in OBSAI mode, and the number of Hyperframes in CPRI mode, that result in state ST3 (FRAME_SYNC). Range: 0 to 65,535

7.4.4.3 RM FSM Unsync Count Configuration

Defines the threshold values to exit the ST2 or ST3 states.

unsync_t - 16 bits

Threshold value for consecutive invalid blocks of bytes that result in state ST0. Range: 0 to 65,535

frame_unsync_t - 16 bits

Threshold value for consecutive invalid message groups that result in state ST1. Range: 0 to 65,535

7.4.4.4 Clock Detection Configuration

The clock monitor register provides programmability of the clock detection circuit.

wd_en – one bit

Enables the clock detect watch dog timer.

- 0: Clock detect watch dog timer Disabled
- 1: Clock detect watch dog timer Enabled

cq_en - one bit

Enables the clock quality circuit.

- 0: Clock quality circuit Disabled
- 1: Clock quality circuit Enabled

wd_wrap - eight bits

Defines the wrap value of the clock detection watch-dog circuit.

mon_wrap - 16 bits

Defines the wrap value of the clock monitor used to define clock quality.

7.4.4.5 RM Status

NOTE: When an RM link is disabled (rx_en = 0) all status is set to the reset state.

fifo_ovf - one bit

Active after an RM FIFO overflow:

- 0: No FIFO error
- 1: FIFO Overflow occurred

los - one bit

Active when Rx state machine is in the UNSYNC (ST0) state, inactive otherwise.

num_los_det - one bit

Detects when *num_los* counter has reached the programmable *los_det_thold* limit.

OBSAI:

- 0: After a master frame of no 8b10b errors OR when configuration value *los_det_thold* is written
- 1: When *num_los* counter has receded *los_det_thold* within a master frame

CPRI:

- 0: After a Hyperframe of no 8b10b errors OR when configuration value *los_det_thold* is written
- 1 : When *num_los* counter has receded *los_det_thold* within a Hyperframe

num_loss - 16 bits

Represents the number of los_det (8b10b code violation) occurrences in a Master Frame (OBSAI) or Hyperframe (CPRI). This counter will saturate and hold its value until either cleared by writing the configuration value *los_det_thold* = 0 OR after a Master Frame (OBSAI) or Hyperframe (CPRI) of no 8b10b errors. Range: 0x0000 to 0xffff

lcv_cnt_value - 16 bits

Number of Line Code Violations counted since last cleared and enabled. Range: 0x0000 to 0xffff LCVs

loc - one bit

The clock watch dog circuit detected a missing clock

- 0: RM clock watch dog not enabled or not missing a clock
- 1: RM clock watch dog detected a missing a clock

clk_qual - 16 bits

A value that represents the quality (relative frequency) of the received SerDes clock. Range: 0x0000 to 0xffff

7.5 TM (TX Mac)

The Tx Mac block provides interconnect from the AIF2 IP block to the SerDes macro transmit interface of the external SerDes module. The Tx Mac function is implemented as six identical blocks with six separate data paths. Each instantiation of the Tx Mac is provided a separate AT Module Interface for system synchronization

The Tx Mac (TM) block supports the data path connection to a SerDes module transmit link of AIF2 IP block. The Tx Mac converts a four byte wide parallel stream into a two byte wide parallel stream in the system clock domain of the transmitted signal that is clocked directly into the SerDes module. The Tx Mac also provides frame formatting and synchronization, and serial coding of either an OBSAI or CPRI signal.

7.5.1 Transmit Frame Synchronization and Formatting

Transmit Frame Synchronization is done by a Transmit Frame Control logic block that controls the read operation of the Transmit FIFO. Transmit data is read out of the FIFO two bytes at a time at the programmed link rate for the link. The Transmit Frame Control Logic receives a frame sync signal from the AT block that is used to align the state machine of the Frame Control logic with the transmit frame timing. The Delta pulse received from the AT module is a single pulse generated in the byte clock domain that indicates the Delay from the RP1 frame sync signal within the AT block for the transmission of the frame to occur.

The variable Delta is contained in a register within the AT block and the Delta pulse is generated by the RP1 system timing clock source. The AT block must convert the Delta pulse into the system clock domain before it can be used to transmit a frame. The frame synchronization logic contains a finite state machine that maintains the frame format of the transmitted signal in clock ticks. Because the output of the FIFO is used to drive the two-byte bus interface for the transmit SerDes macro, the system clock ticks are used in a manner such that for an 8x rate, every clock tick is used, for a 4x rate every other clock tick is used, and for a 2x rate one of four clock ticks are used. A frame byte counter for an entire frame is also maintained on these clock ticks. Because frame synchronization is done within the AT block in the RP1 system clock domain (61.44 MHz), the Delta pulse and the frame byte counter are only as precise as the timing difference between the system clock domain and the RP1 system clock domain will allow. The precision of this alignment is done at the frequency of the system clock divided by the number of clock ticks times the number of synchronization stages required for the Delta pulse to synchronize to the system clock.

7.5.1.1 Frame Construction

Functionally, the transmit data is byte aligned to the frame structure of the transmitted signal. The frame is established by the insertion of K characters by Frame Control logic. The K characters indicate start of frame or idle conditions. Data is optionally scrambled for OBSAI 8x rate only, but k characters are never scrambled. The scrambler is placed into an initialization phase with an initial seed value. The scrambler operation and seed value initialization is further detailed in a separate section of this document. The resultant two byte wide data path constitutes the transmission path for a frame.

7.5.1.2 Transmit Frame Control

The Transmit Frame Control logic contains a finite state machine that manages the operation of the transmit data path. The state machine performs similar transitions in both OBSAI and CPRI modes. The basic operation in both modes provides a status of frame transmission on the transmitted signal that is used to manage the control signals of the data path.

Figure 7-11. Transmit State Machine

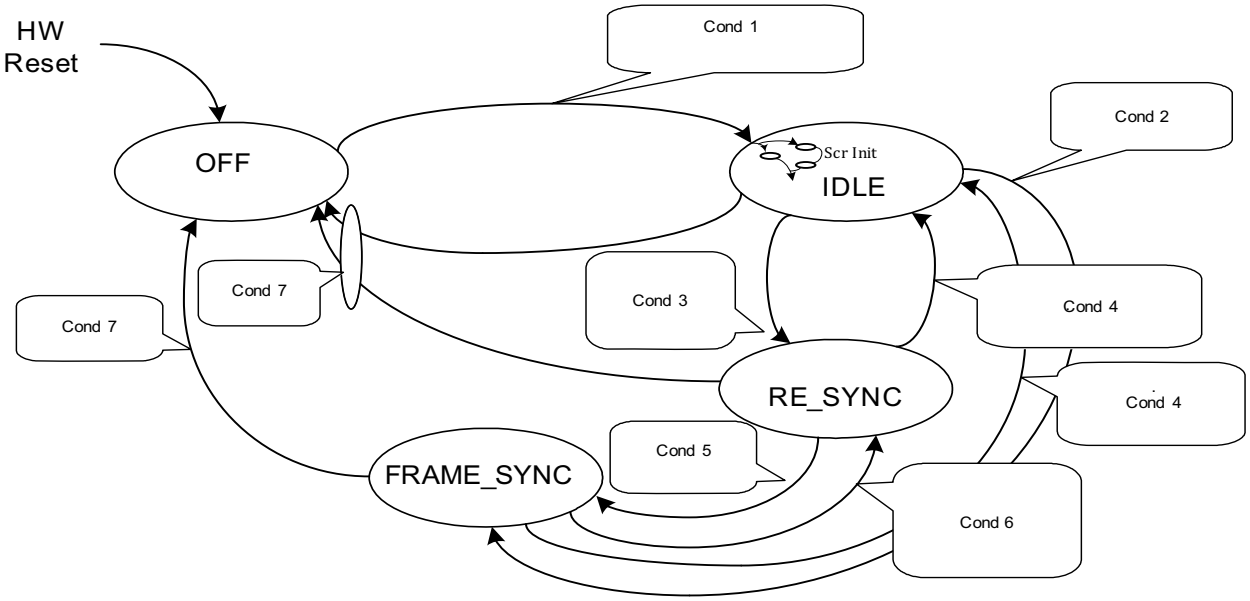


Table 7-8. Frame Sync State Machine Transition Conditions

Transition	Condition
Cond 1	TM Enable and not (RM Loss of Signal and LOS_EN)
Cond 2	Delta Pulse and FIFO Ready and (not Scr_En or (Scr_En and RM_Seed_Acq and RM_IDLE_ACK))
Cond 3	Delta Pulse and FIFO Not Ready and (not Scr_En or (Scr_En and RM_Seed_Acq and RM_IDLE_ACK))
Cond 4	(Scr_En and not (RM_Seed_Acq and RM_IDLE_ACK))
Cond 5	FIFO Ready and Delta Pulse
Cond 6	Transmit Data Frame Mismatch or FIFO Starve or Delta Pulse Inactive/Modified or TM_FLUSH
Cond 7	TM Disable or (RM Loss of Signal and LOS_En)

In general, the state machine resets to an ‘Off’ state that indicates the data path is not yet operational. Upon enabling of the TM block, the state machine transitions to the ‘IDLE’ state in which it forces the Tx Mac to generate k characters, or in the 8x case, the scrambler seed protocol. Upon reception of the ‘Delta’ pulse (for 8x case, the seed protocol must be satisfied), the state machine transitions to the ‘resync’ state that initiates the construction of empty frames and messages that are transmitted. The state machine waits for the FIFO to be ready so that it can transition to the ‘Frame Sync’ state for normal transmit operation. The ‘Frame Sync’ state is exited upon FIFO starvation, a missing or changed ‘Delta’ pulsed, an indication that the scrambler seed protocol had been detected by the RM block, or upon a disabling of the TM block.

7.5.2 Transmit FIFO

The Transmit FIFO of each Tx Mac link provides an interface between the steady transmit rate of the two byte interface of the SerDes macro transmit interface and the gapped rate of the four byte parallel stream input that is received from the CO block of the antenna interface data path. The write operation to the FIFO is supported by control logic that emulates the configured link rate by asserting back pressuring with the read enable signal back to the CO block. The FIFO is completely synchronous on both the read and the write port.

The Transmit FIFO contains dedicated signals that provide k character indication for each byte in the FIFO. The k character indications are used to enable the 8b/10 encoding logic to insert a k character symbol for the data byte for which the indication is active. The Transmit FIFO terminates alignment side band signals and creates k character indications for each byte stored in the FIFO. These side band signals indicate for OBSAI:

- 10 msec Master Frame Boundary
- Message Group Boundary

and for CPRI:

- 10 msec Frame Boundary
- Hyper-frame Boundary
- Basic Frame Boundary
- Control Word Packet Boundary

The FIFO begins to fill upon a reception of master frame boundary, and asserts back pressure to avoid overflow. The FIFO begins to empty once the Delta pulse is received. The empty rate is a power of two divisor of the system clock depending if the link rate is set to 2x, 4x or 8x. The FIFO continuously empties at a steady rate until the transmit state machine initiates a resynchronization of the data path, or the FIFO indicates a starvation state to the transmit state machine, in which case a resynchronization of the data path is also resynchronized.

7.5.3 Tx Mac Programming

7.5.3.1 TM State Machine Control Register (AIF2_TM_CTRL[0-5])

LOS_EN – Bit 3

Allows Loss of Signal from the RM to cause the TM state machine to transition to the OFF state.

- 1'b0: Loss of Signal Disabled
- 1'b1: Loss of Signal Enabled

TM RESYNC – Bit 2

Forces the Frame Sync state machine to transition from the FRAME_SYNC state to the RESYNC state and remain in the RESYNC state until the bit is cleared. If the state machine is in either the OFF state or the IDLE state, this bit is ignored until the state machine reaches the RESYNC state.

TM IDLE – Bit 1

Forces the Frame Sync state machine to transition form either the FRAME_SYNC or RESYNC states to the IDLE state and remain in the IDLE state until the bit is cleared. If the state machine is in the OFF state, this bit is ignored until the state machine reaches the IDLE state.

TM Flush – Bit 0

Instructs the TM link to flush the FIFO and force a TM Fail condition on the link. When set high, the State machine is forced into the RE_SYNC state.

7.5.3.2 TM Scrambler Control Register (AIF2_TM_SCR_CTRL[0-5])

The Scrambler Configuration Register contains the seed initialization vector for the LFSR scrambler utilized when scrambling is enabled is OBSAI 8x mode, and the scrambler enable bit. This configuration register should only be updated when the Frame Sync state machine is disabled.

SCR_EN - Bit 7

Enables the scrambler for OBSAI 8x speed link operation in the transmit data path.

1'b0: Scrambler Disabled

1'b1: Scrambler Enabled

Seed Value - Bits 6:0

The seed value is used to initialize the transmit scrambler circuit.

7.5.3.3 TM CPRI L1 Inband Configuration Register (AIF2_TM_L1_CFG[0-5])

The L1 Inband Configuration Register allows programming of the L1 inband control signals for CPRI mode

L1_Inband – Bits 4:0

- Bit 4: LOF
- Bit 3: LOS
- Bit 2: SDI
- Bit 1: RAI
- Bit 0: Reset

7.5.3.4 TM CPRI L1 Inband Enable Register (AIF2_TM_L1_EN[0-5])

The L1 Inband Enable Register allows hardware control of the L1 inband control signals for CPRI mode. A ‘1’ for each bit indicates the hardware control is enabled. Each enable bit is a gate on an input condition that could affect an output condition. The nomenclature is defined as ‘TXSIGNAL_RXCOND_EN. The term ‘ERR’ indicates the error had been determined by the RM, while the term ‘RX’ refers to the actual L1 inband signal received by the RM.

L1_Inband – Bits 8:0

- Bit 8: LOS_LOSERR_EN
- Bit 7: LOS_LOSRX_EN
- Bit 6: LOF_LOFERR_EN
- Bit 5: LOF_LOFRX_EN
- Bit 4: RAI_LOSERR_EN
- Bit 3: RAI_LOSRX_EN
- Bit 2: RAI_LOFERR_EN
- Bit 1: RAI_LOFRX_EN
- Bit 0: RAI_RAIRX_EN

7.5.3.5 TM CPRI L1 Inband LOSERR Link Select Register (AIF2_TM_LOSERR[0-5])

Selects which RM link is used to drive the LOSERR condition to determine transmit L1 Inband signaling.

RM_Link_LOSERR – Bits 3:0

- 3'x0: RM Link LOSERR 0
- 3'x1: RM Link LOSERR 1
- 3'x2: RM Link LOSERR 2
- 3'x3: RM Link LOSERR 3
- 3'x4: RM Link LOSERR 4
- 3'x5: RM Link LOSERR 5

7.5.3.6 TM CPRI L1 Inband LOFERR Link Select Register (AIF2_TM_LOFERR[0-5])

Selects which RM link is used to drive the LORERR condition to determine transmit L1 Inband signaling

RM_Link_LOFERR – Bits 3:0

- 3'x0: RM Link 0
- 3'x1: RM Link 1
- 3'x2: RM Link 2
- 3'x3: RM Link 3
- 3'x4: RM Link 4
- 3'x5: RM Link 5

7.5.3.7 TM CPRI L1 Inband LOSRx Link Select Register (AIF2_TM_LOSRx[0-5])

Selects which RM link is used to drive the LOSRx condition to determine transmit L1 Inband signaling

RM_Link_LOSRx – Bits 3:0

- 3'x0: RM Link LOSRx 0
- 3'x1: RM Link LOSRx 1
- 3'x2: RM Link LOSRx 2
- 3'x3: RM Link LOSRx 3
- 3'x4: RM Link LOSRx 4
- 3'x5: RM Link LOSRx 5

7.5.3.8 TM CPRI L1 Inband LOFRx Link Select Register (AIF2_TM_LOFRx[0-5])

Selects which RM link is used to drive the LOFRx condition to determine transmit L1 Inband signaling

RM_Link_LOFRx – Bits 3:0

- 3'x0: RM Link LOFRx 0
- 3'x1: RM Link LOFRx 1
- 3'x2: RM Link LOFRx 2
- 3'x3: RM Link LOFRx 3
- 3'x4: RM Link LOFRx 4
- 3'x5: RM Link LOFRx 5

7.5.3.9 TM CPRI L1 Inband RAIRx Link Select Register (AIF2_TM_RAIRx[0-5])

Selects which RM link is used to drive the RAIRx condition to determine transmit L1 Inband signaling

RM_Link_RAIRx – Bits 3:0

- 3'x0: RM Link RAIRx 0
- 3'x1: RM Link RAIRx 1
- 3'x2: RM Link RAIRx 2
- 3'x3: RM Link RAIRx 3
- 3'x4: RM Link RAIRx 4
- 3'x5: RM Link RAIRx 5

7.5.3.10 TM CPRI Pointer P Configuration Register (AIF2_TM_PTRP[0-5])

PTR_P - Bits 7:0

Contains the Pointer P value

7.5.3.11 TM CPRI Startup Configuration Register (AIF2_TM_STRT[0-5])

STARTUP - Bits 7:0

Contains the Startup Value

7.5.3.12 TM CPRI Protocol Version Configuration (AIF2_TM_PROT[0-5])

PROT_VERS - Bits 7:0

Contains the Protocol Version Value

7.6 CI and CO (CPRI Input and Output Data Module)

A CPRI conversion function is required for CPRI data frames between the physical interface and the system for AIF2 to support CPRI. The conversion relieves the system of interpreting interleaved I and Q data bits that exist in the CPRI mapping of antenna carriers within CPRI frames. The CPRI data format interleaves IQ data on a bit-per-bit basis. It also supports I and Q widths of seven or eight bits for uplink and 15 or 16 bits for downlink. As such, CPRI Conversion performs the following data manipulations:

- Provides Conversion between CPRI frame format and Internal Data Format
- IQ data bit interleave/deinterleave within antenna carrier samples
- Byte aligns seven and 15-bit IQ data to internal 32-bit data bus
- For seven-bit format, IQ data sign extension to yield eight bits
- For 15-bit format, IQ data sign extension to yield 16 bits
- Supports pass-through of 'r' bits in eight-bit UL and 16-bit DL at 2x link rate
- Provides 'I' or 'Q' saturation for the transmitter

7.6.1 CPRI Data Formats

The supported mapping of an AxC container within a CPRI basic frame changes per the number of data bits used as shown in Table 7-9.

Table 7-9. Supported CPRI IQ Sample Widths

Direction	Bits	Packing Option	Packing Description
Downlink	16	Packed	w/ "r" bits at the end of basic frame
Downlink	15	Packed	no "r" bits
Uplink	8	Packed	w/ "r" bits at the end of basic frame
Uplink	7	Flexible	w/ "r,r" after every AxC container

The input data format for all of the CPRI line rates are shown in Table 7-10. The CPRI internal data formats for these are shown in the following figures. In all of the examples, the CPRI Data Format is expanded, from 32 to 34 bytes for 2X links, from 64 to 68 bytes for 4X links, and from 128 to 136 bytes for 8x links. In 8- and 16-bit modes, the extra bits are all reserved. Note that the 8- and 16-bit formats always have one less AxC stream than the equivalent seven and 15-bit formats (that is, 7 vs 8 for 2X; and 15 vs 16 for 4X). In all of the figures, the data is received from top left down and to the right. Table 7-10 is fully applied for WCDMA case. For OFDM (LTE, WiMAX, ...), both Uplink and Downlink can support 15 bit and 16 bit mode.

Table 7-10. Supported CPRI Line Rates and Data Widths

Direction	Link Rate	Bits
Uplink	2X	7
Uplink	2X	8
Uplink	4X	7
Uplink	4X	8
Uplink	8X	7
Uplink	8X	8
Downlink	2X	15
Downlink	2X	16
Downlink	4X	15
Downlink	4X	16
Downlink	8X	15
Downlink	8X	16

Figure 7-12. CPRI Data Format (2x link - 15-bit DL, 7-bit UL)

		CW	A0	A1	A2	A3	A4	A5	A6	A7
			16 words = 32 bytes							
DL	LSB	Control Word	Io,o	Io,8	I1,1	I1,9	I2,2	I2,10	I3,3	I3,11	I4,4	I4,12	I5,5	I5,13	I6,6	I6,14	I7,7
			Qo,o	Qo,8	Q1,1	Q1,9	Q2,2	Q2,10	Q3,3	Q3,11	Q4,4	Q4,12	Q5,5	Q5,13	Q6,6	Q6,14	Q7,7
			Io,1	Io,9	I1,2	I1,10	I2,3	I2,11	I3,4	I3,12	I4,5	I4,13	I5,6	I5,14	I6,7	I7,o	I7,8
			Qo,1	Qo,9	Q1,2	Q1,10	Q2,3	Q2,11	Q3,4	Q3,12	Q4,5	Q4,13	Q5,6	Q5,14	Q6,7	Q7,o	Q7,8
			Io,2	Io,10	I1,3	I1,11	I2,4	I2,12	I3,5	I3,13	I4,6	I4,14	I5,7	I6,o	I6,8	I7,1	I7,9
			Qo,2	Q0,10	Q1,3	Q1,11	Q2,4	Q2,10	Q3,5	Q3,13	Q4,6	Q4,14	Q5,7	Q6,o	Q6,8	Q7,1	Q7,9
			Io,3	Io,11	I1,4	I1,12	I2,5	I2,13	I3,6	I3,14	I4,7	I5,0	I5,8	I6,1	I6,9	I7,2	I7,10
	MSB		Qo,3	Q0,11	Q1,4	Q1,12	Q2,5	Q2,13	Q3,6	Q3,14	Q4,7	Q5,0	Q5,8	Q6,1	Q6,9	Q7,2	Q7,10
	LSB	Control Word	Io,4	Io,12	I1,5	I1,13	I2,6	I2,14	I3,7	I4,o	I4,8	I5,1	I5,9	I6,2	I6,10	I7,3	I7,11
			Qo,4	Q0,12	Q1,5	Q1,13	Q2,6	Q2,14	Q3,7	Q4,o	Q4,8	Q5,1	Q5,9	Q6,2	Q6,10	Q7,3	Q7,11
			Io,5	Io,13	I1,6	I1,14	I2,7	I3,o	I3,8	I4,1	I4,9	I5,2	I5,10	I6,3	I6,11	I7,4	I7,12
			Qo,5	Q0,13	Q1,6	Q1,14	Q2,7	Q3,o	Q3,8	Q4,1	Q4,9	Q5,2	Q5,10	Q6,3	Q6,11	Q7,4	Q7,12
			Io,6	Io,14	I1,7	I2,o	I2,8	I3,1	I3,9	I4,2	I4,10	I5,3	I5,11	I6,4	I6,12	I7,5	I7,13
			Qo,6	Q0,14	Q1,7	Q2,o	Q2,8	Q3,1	Q3,9	Q4,2	Q4,10	Q5,3	Q5,11	Q6,4	Q6,10	Q7,5	Q7,13
			Io,7	I1,0	I1,8	I2,1	I2,9	I3,2	I3,10	I4,3	I4,11	I5,4	I5,12	I6,5	I6,13	I7,6	I7,14
	MSB		Qo,7	Q1,0	Q1,8	Q2,1	Q2,9	Q3,2	Q3,10	Q4,3	Q4,11	Q5,4	Q5,12	Q6,5	Q6,13	Q7,6	Q7,14

UL	LSB	Control Word	Io,o	I'o,1	I1,1	I'1,2	I2,2	I'2,3	I3,3	I'3,4	I4,4	I'4,5	I5,5	I'5,6	I6,6	r	I'7,o
			Qo,o	Q'o,1	Q1,1	Q'1,2	Q2,2	Q'2,3	Q3,3	Q'3,4	Q4,4	Q'4,5	Q5,5	Q'5,6	Q6,6	r	Q'7,o
			Io,1	I'o,2	I1,2	I'1,3	I2,3	I'2,4	I3,4	I'3,5	I4,5	I'4,6	I5,6	r	I'6,o	I7,o	I'7,1
			Qo,1	Q'o,2	Q1,2	Q'1,3	Q2,3	Q'2,4	Q3,4	Q'3,5	Q4,5	Q'4,6	Q5,6	r	Q'6,o	Q7,o	Q'7,1
			Io,2	I'o,3	I1,3	I'1,4	I2,4	I'2,5	I3,5	I'3,6	I4,6	r	I'5,o	I6,0	I'6,1	I7,1	I'7,2
			Qo,2	Q'o,3	Q1,3	Q'1,4	Q2,4	Q'2,5	Q3,5	Q'3,6	Q4,6	r	Q'5,o	Q6,0	Q'6,1	Q7,1	Q'7,2
			Io,3	I'o,4	I1,4	I'1,5	I2,5	I'2,6	I3,6	r	I'4,o	I5,o	I'5,1	I6,1	I'6,2	I7,2	I'7,3
	MSB		Qo,3	Q'o,4	Q1,4	Q'1,5	Q2,5	Q'2,6	Q3,6	r	Q'4,o	Q5,o	Q'5,1	Q6,1	Q'6,2	Q7,2	Q'7,3
	LSB	Control Word	Io,4	I'o,5	I1,5	I'1,6	I2,6	r	I'3,o	I4,o	I'4,1	I5,1	I'5,2	I6,2	I'6,3	I7,3	I'7,4
			Qo,4	Q'o,5	Q1,5	Q'1,6	Q2,6	r	Q'3,o	Q4,o	Q'4,1	Q5,1	Q'5,2	Q6,2	Q'6,3	Q7,3	Q'7,4
			Io,5	I'o,6	I1,6	r	I'2,o	I3,o	I'3,1	I4,1	I'4,2	I5,2	I'5,3	I6,3	I'6,4	I7,4	I'7,5
			Qo,5	Q'o,6	Q1,6	r	Q'2,o	Q3,o	Q'3,1	Q4,1	Q'4,2	Q5,2	Q'5,3	Q6,3	Q'6,4	Q7,4	Q'7,5
			Io,6	r	I'1,o	I2,0	I'2,1	I3,1	I'3,2	I4,2	I'4,3	I5,3	I'5,4	I6,4	I'6,5	I7,5	I'7,6
			Qo,6	r	Q'1,o	Q2,0	Q'2,1	Q3,1	Q'3,2	Q4,2	Q'4,3	Q5,3	Q'5,4	Q6,4	Q'6,5	Q7,5	Q'7,6
			I'o,o	I1,o	I'1,1	I2,1	I'2,2	I3,2	I'3,3	I4,3	I'4,4	I5,4	I'5,5	I6,5	I'6,6	I7,6	r
	MSB		Q'o,o	Q1,o	Q'1,1	Q2,1	Q'2,2	Q3,2	Q'3,3	Q4,3	Q'4,4	Q5,4	Q'5,5	Q6,5	Q'6,6	Q7,6	r

Figure 7-13. CPRI Data Format (5x link 7-bit UL)

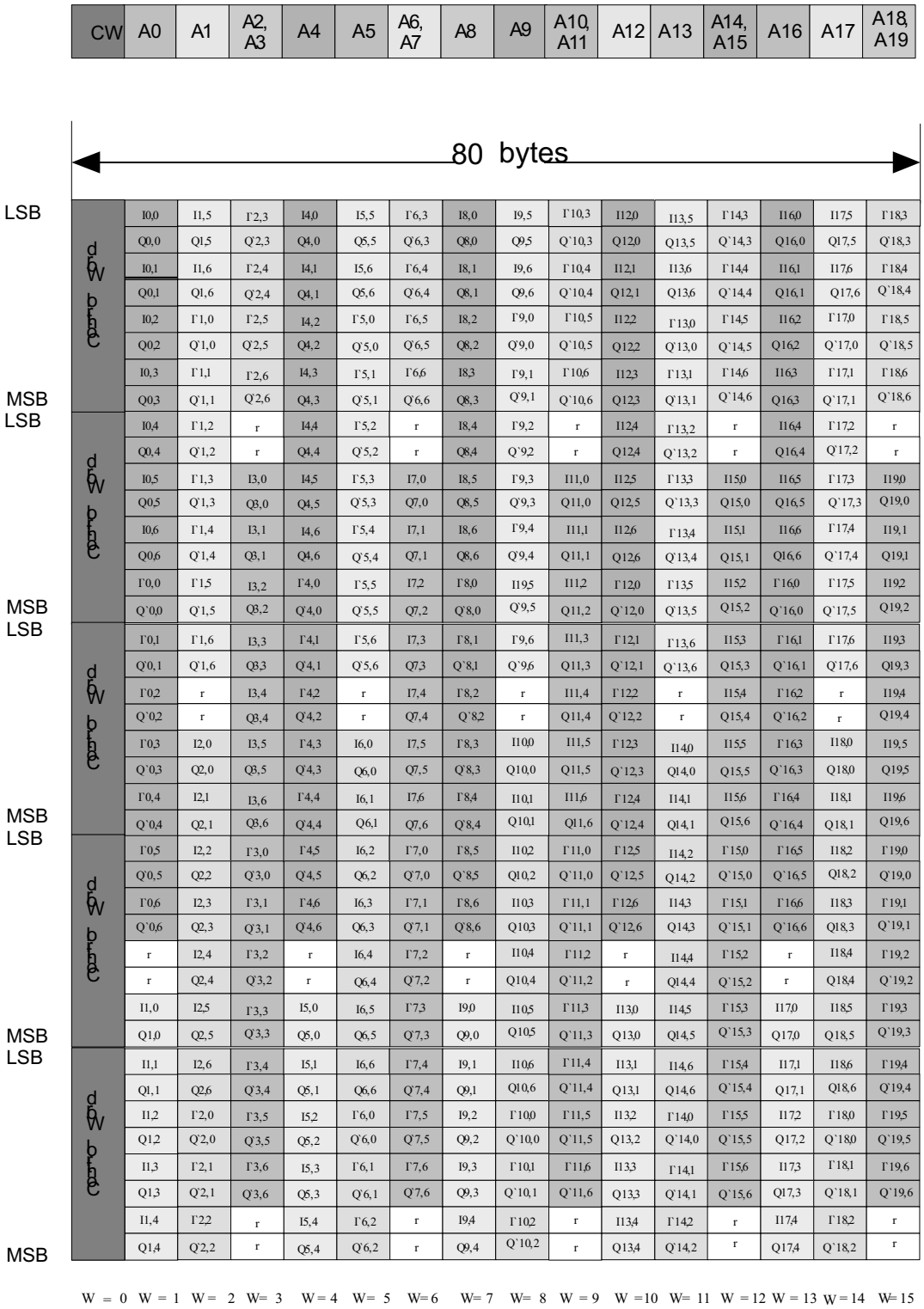


Figure 7-14. CPRI Data Format (5x link 16-bit DL)

CW	A0	A1	A2	A3, A4	A5	A6	A7	A8, A9	A10	A11	A12	A13, A14	A15	A16	A17

80 bytes

LSB	Contrd Word	I0,0	I1,4	I2,8	I3,12	I5,0	I6,4	I7,8	I8,12	I10,0	I11,4	I12,8	I13,12	I15,0	I16,4	I17,8
		Q0,0	Q1,4	Q2,8	Q3,12	Q5,0	Q6,4	Q7,8	Q8,12	Q 10,0	Q 11,4	Q 12,8	Q13,12	Q 15,0	Q16,4	Q17,8
		I0,1	I1,5	I2,9	I3,13	I5,1	I6,5	I7,9	I8,13	I10,1	I11,5	I12,9	I13,13	I15,1	I16,5	I17,9
		Q 0,1	Q1,5	Q2,9	Q3,13	Q5,1	Q6,5	Q7,9	Q 8,13	Q 10,1	Q 11,5	Q 12,9	Q13,13	Q 15,1	Q 16,5	Q 17,9
		I0,2	I1,6	I2,10	I3,14	I5,2	I6,6	I7,10	I8,14	I10,2	I11,6	I12,10	I13,14	I15,2	I16,6	I17,10
		Q 0,2	Q1,6	Q2,10	Q3, 14	Q5,2	Q6,6	Q 7,10	Q 8,14	Q 10,2	Q 11,6	Q 12,10	Q13, 14	Q 15,2	Q 16,6	Q17, 10
		I0,3	I1,7	I2,11	I3,15	I5,3	I6,7	I7, 11	I8,15	I10,3	I11,7	I12, 11	I13, 15	I15,3	I16,7	I17, 11
MSB		Q 0,3	Q1,7	Q 2,11	Q 3,15	Q5,3	Q6,7	Q 7,11	Q 8,15	Q 10,3	Q 11,7	Q 12,11	Q13, 15	Q 15,3	Q 16,7	Q17, 11
LSB	Contrd Word	I0,4	I1,8	I2,12	I4,0	I5,4	I6,8	I7,12	I9,0	I10,4	I11,8	I12,12	I14,0	I15,4	I16,8	I17, 12
		Q0,4	Q1,8	Q2, 12	Q4,0	Q5,4	Q6,8	Q7, 12	Q9,0	Q 10,4	Q 11,8	Q12, 12	Q 14,0	Q 15,4	Q16, 8	Q 17,12
		I0,5	I1,9	I2,13	I4,1	I5,5	I6,9	I7,13	I9,1	I10,5	I11,9	I12, 13	I14,1	I15,5	I16,9	I17, 13
		Q 0,5	Q1,9	Q 2,13	Q4,1	Q5,5	Q6,9	Q 7,13	Q9,1	Q 10,5	Q 11,9	Q12, 13	Q 14,1	Q 15,5	Q 16,9	Q17, 13
		I0,6	I1,10	I2,14	I4,2	I5,6	I6,10	I7,14	I9,2	I10,6	I11, 10	I12, 14	I14,2	I15,6	I16, 10	I17,14
		Q 0,6	Q 1,10	Q2,14	Q4,2	Q5,6	Q6,10	Q 7,14	Q9,2	Q 10,6	Q11, 10	Q 12,14	Q 14,2	Q15,6	Q16, 10	Q17, 14
		I0,7	I1,11	I2,15	I4,3	I5,7	I6,11	I7,15	I9,3	I10,7	I11, 11	I12, 15	I14,3	I15,7	I16, 11	I17, 15
MSB		Q 0,7	Q 1,11	Q 2,15	Q4,3	Q5,7	Q6,11	Q 7,15	Q9,3	Q 10,7	Q11, 11	Q 12,15	Q 14,3	Q 15,7	Q16, 11	Q17, 15
LSB	Contrd Word	I0,8	I1,12	I3,0	I4,4	I5,8	I6,12	I8,0	I9,4	I10,8	I11, 12	I13,0	I14,4	I15,8	I16, 12	r
		Q0,8	Q1, 12	Q3,0	Q4,4	Q5,8	Q6, 12	Q8,0	Q9,4	Q 10,8	Q11, 12	Q 13,0	Q 14,4	Q 15,8	Q 16,12	r
		I0,9	I1,13	I3,1	I4,5	I5,9	I6,13	I8,1	I9,5	I10,9	I11, 13	I13,1	I14,5	I15,9	I16, 13	r
		Q 0,9	Q 1,13	Q3,1	Q4,5	Q5,9	Q6,13	Q 8,1	Q9,5	Q 10,9	Q11, 13	Q 13,1	Q 14,5	Q 15,9	Q16, 13	r
		I0, 10	I1,14	I3,2	I4,6	I5,10	I6,14	I8,2	I9,6	I10 ,10	I11, 14	I13,2	I14,6	I15, 10	I16, 14	r
		Q0, 10	Q 1,14	Q3,2	Q4,6	Q 5,10	Q6,14	Q8,2	Q9,6	Q10, 10	Q11, 14	Q13,2	Q 14,6	Q15, 10	Q16, 14	r
		I0,11	I1,15	I3,3	I4,7	I5,11	I6, 15	I8,3	I9,7	I10, 11	I11, 15	I13,3	I14,7	I15, 11	I16, 15	r
MSB		Q0, 11	Q 1,15	Q3,3	Q4,7	Q5, 11	Q6,15	Q8, 3	Q9,7	Q10 ,11	Q11, 15	Q13,3	Q 14,7	Q15, 11	Q16, 15	r
LSB	Contrd Word	I0, 12	I2,0	I3,4	I4,8	I5,12	I7,0	I8,4	I9,8	I10,12	I12,0	I13,4	I14,8	I15, 12	I17,0	r
		Q0,12	Q2,0	Q3,4	Q4,8	Q 5,12	Q7,0	Q8,4	Q9,8	Q10, 12	Q 12,0	Q 13,4	Q 14,8	Q15, 12	Q17,0	r
		I0, 13	I'2, 1	I3,5	I4,9	I5,13	I7,1	I8,5	I9,9	I10,13	I12,1	I13,5	I14,9	I15, 13	I17,1	r
		Q0, 13	Q2,1	Q3,5	Q4,9	Q 5,13	Q7,1	Q8,5	Q9,9	Q10, 13	Q 12,1	Q 13,5	Q 14,9	Q15, 13	Q 17,1	r
		I0, 14	I2,2	I3,6	I4,10	I5,14	I7,2	I8,6	I9,10	I10, 14	I12,2	I13,6	I14, 10	I15, 14	I17,2	r
		Q0, 14	Q2,2	Q3,6	Q 4,10	Q 5,14	Q7,2	Q8,6	Q 9,10	Q10, 14	Q 12,2	Q13,6	Q14, 10	Q 15,14	Q 17,2	r
		I0,15	I2,3	I3,7	I4,11	I5,15	I7,3	I8,7	I9,11	I10, 15	I12,3	I13,7	I14, 11	I15, 15	I17,3	r
MSB		Q0, 15	Q2,3	Q3,7	Q 4,11	Q 5,15	Q7,3	Q8,7	Q 9,11	Q10, 15	Q 12,3	Q13,7	Q14, 11	Q15, 15	Q 17,3	r
LSB	Control Word	I1,0	I2,4	I3,8	I4,12	I6,0	I7,4	I8,8	I9,12	I11,0	I12,4	I13,8	I14,12	I16,0	I17,4	r
		Q1,0	Q2,4	Q3,8	Q 4,12	Q6,0	Q7,4	Q8,8	Q9, 12	Q 11,0	Q 12,4	Q 13,8	Q14, 12	Q 16,0	Q17,4	r
		I1,1	I2,5	I3,9	I4,13	I6,1	I7,5	I8,9	I9,13	I11,1	I12,5	I13,9	I14,13	I16,1	I17,5	r
		Q 1,1	Q2,5	Q3,9	Q 4,13	Q6,1	Q7,5	Q8,9	Q 9,13	Q 11,1	Q 12,5	Q 13,9	Q14,13	Q 16,1	Q 17,5	r
		I1,2	I2,6	I3,10	I4,14	I6,2	I7,6	I8,10	I9,14	I11,2	I12,6	I13,10	I14,14	I16,2	I17,6	r
		Q 1,2	Q2,6	Q3,10	Q 4,14	Q6,2	Q7,6	Q 8,10	Q 9,14	Q 11,2	Q 12,6	Q 13,10	Q14,14	Q16,2	Q 17,6	r
		I1,3	I2,7	I3,11	I4,15	I6,3	I7,7	I8, 11	I9,15	I11,3	I12,7	I13, 11	I14,15	I16,3	I17,7	r
MSB		Q 1,3	Q2,7	Q 3,11	Q 4,15	Q6,3	Q7,7	Q 8,11	Q 9,15	Q 11,3	Q 12,7	Q 13,11	Q14, 15	Q 16,3	Q 17,7	r

W = 0 W = 1 W = 2 W= 3 W = 4 W= 5 W= 6 W= 7 W= 8 W = 9 W =10 W= 11 W = 12 W = 13 W = 14 W= 15

Figure 7-15. CPRI Data Format (4x link – 8-bit UL)

	CW	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14
		16 words = 64 bytes														
LSB	Control Word	I0,0	I1,0	I2,0	I3,0	I4,0	I5,0	I6,0	I7,0	I8,0	I9,0	I10,0	I11,0	I12,0	I13,0	I14,0
		Q0,0	Q1,0	Q2,0	Q3,0	Q4,0	Q5,0	Q6,0	Q7,0	Q8,0	Q9,0	Q10,0	Q11,0	Q12,0	Q13,0	Q14,0
		I0,1	I1,1	I2,1	I	I	I	I	I	I	I	I	I	I	I	I
UL		Q0,1	Q1,1	Q2,1	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
		I0,2	I1,2	I2,2	I	I	I	I	I	I	I	I	I	I	I	I
		Q0,2	Q1,2	Q2,2	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
		I0,3	I1,3	I2,3	I	I	I	I	I	I	I	I	I	I	I	I
MSB		Q0,3	Q1,3	Q2,3	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
LSB	Control Word	I0,4	I1,4	I2,4	I	I	I	I	I	I	I	I	I	I	I	I
		Q0,4	Q1,4	Q2,4	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
		I0,5	I1,5	I2,5	I	I	I	I	I	I	I	I	I	I	I	I
		Q0,5	Q1,5	Q2,5	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
		I0,6	I1,6	I2,6	I	I	I	I	I	I	I	I	I	I	I	I
		Q0,6	Q1,6	Q2,6	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
		I0,7	I1,7	I2,7	I3,7	I4,7	I5,7	I6,7	I7,7	I8,7	I9,7	I10,7	I11,7	I12,7	I13,7	I14,7
MSB		Q0,7	Q1,7	Q2,7	Q3,7	Q4,7	Q5,7	Q6,7	Q7,7	Q8,7	Q9,7	Q10,7	Q11,7	Q12,7	Q13,7	Q14,7
LSB	Control Word	I'0,0	I'1,0	I'2,0	I'3,0	I'4,0	I'5,0	I'6,0	I'7,0	I'8,0	I'9,0	I'10,0	I'11,0	I'12,0	I'13,0	I'14,0
		Q'0,0	Q'1,0	Q'2,0	Q'3,0	Q'4,0	Q'5,0	Q'6,0	Q'7,0	Q'8,0	Q'9,0	Q'10,0	Q'11,0	Q'12,0	Q'13,0	Q'14,0
		I'0,1	I'1,1	I'2,1	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'
		Q'0,1	Q'1,1	Q'2,1	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'
		I'0,2	I'1,2	I'2,2	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'
		Q'0,2	Q'1,2	Q'2,2	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'
		I'0,3	I'1,3	I'2,3	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'
MSB		Q'0,3	Q'1,3	Q'2,3	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'
LSB	Control Word	I'0,4	I'1,4	I'2,4	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'
		Q'0,4	Q'1,4	Q'2,4	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'
		I'0,5	I'1,5	I'2,5	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'
		Q'0,5	Q'1,5	Q'2,5	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'
		I'0,6	I'1,6	I'2,6	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'	I'
		Q'0,6	Q'1,6	Q'2,6	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'	Q'
		I'0,7	I'1,7	I'2,7	I'3,7	I'4,7	I'5,7	I'6,7	I'7,7	I'8,7	I'9,7	I'10,7	I'11,7	I'12,7	I'13,7	I'14,7
MSB		Q'0,7	Q'1,7	Q'2,7	Q'3,7	Q'4,7	Q'5,7	Q'6,7	Q'7,7	Q'8,7	Q'9,7	Q'10,7	Q'11,7	Q'12,7	Q'13,7	Q'14,7

Figure 7-16. CPRI Internal Uplink Data Format (2x Link / 8-bit Data)

	CW	A0	A1	A2	A3	A4	A5	A6	Reserve	
		34 bytes								
MSB	Control Word	Io,7	I'o,7	I1,7	I'1,7	I2,7	I'2,7	I3,7	I3,7	I4,7	I'4,7	I5,7	I5,7	I6,7	I'6,7	r	r
		Io,6	I'o,6	I1,6	I'1,6	I2,6	I'2,6	I3,6	I3,6	I4,6	I'4,6	I5,6	I5,6	I6,6	I'6,6	r	r
		Io,5	I'o,5	I1,5	I'1,5	I2,5	I'2,5	I3,5	I3,5	I4,5	I'4,5	I5,5	I5,5	I6,5	I'6,5	r	r
		Io,4	I'o,4	I1,4	I'1,4	I2,4	I'2,4	I3,4	I3,4	I4,4	I'4,4	I5,4	I5,4	I6,4	I'6,4	r	r
Y=0		Io,3	I'o,3	I1,3	I'1,3	I2,3	I'2,3	I3,3	I3,3	I4,3	I'4,3	I5,3	I5,3	I6,3	I'6,3	r	r
		Io,2	I'o,2	I1,2	I'1,2	I2,2	I'2,2	I3,2	I3,2	I4,2	I'4,2	I5,2	I5,2	I6,2	I'6,2	r	r
		Io,1	I'o,1	I1,1	I'1,1	I2,1	I'2,1	I3,1	I3,1	I4,1	I'4,1	I5,1	I5,1	I6,1	I'6,1	r	r
UL	LSB	Io,0	I'o,0	I1,0	I'1,0	I2,0	I'2,0	I3,0	I3,0	I4,0	I'4,0	I5,0	I5,0	I6,0	I'6,0	r	r
	MSB	Qo,7	Q'o,7	Q1,7	Q'1,7	Q2,7	Q'2,7	Q3,7	Q3,7	Q4,7	Q'4,7	Q5,7	Q5,7	Q6,7	Q6,7	r	r
		Qo,6	Q'o,6	Q1,6	Q'1,6	Q2,6	Q'2,6	Q3,6	Q3,6	Q4,6	Q'4,6	Q5,6	Q5,6	Q6,6	Q6,6	r	r
		Qo,5	Q'o,5	Q1,5	Q'1,5	Q2,5	Q'2,5	Q3,5	Q3,5	Q4,5	Q'4,5	Q5,5	Q5,5	Q6,5	Q6,5	r	r
		Qo,4	Q'o,4	Q1,4	Q'1,4	Q2,4	Q'2,4	Q3,4	Q3,4	Q4,4	Q'4,4	Q5,4	Q5,4	Q6,4	Q6,4	r	r
Y=1		Qo,3	Q'o,3	Q1,3	Q'1,3	Q2,3	Q'2,3	Q3,3	Q3,3	Q4,3	Q'4,3	Q5,3	Q5,3	Q6,3	Q6,3	r	r
		Qo,2	Q'o,2	Q1,2	Q'1,2	Q2,2	Q'2,2	Q3,2	Q3,2	Q4,2	Q'4,2	Q5,2	Q5,2	Q6,2	Q6,2	r	r
		Qo,1	Q'o,1	Q1,1	Q'1,1	Q2,1	Q'2,1	Q3,1	Q3,1	Q4,1	Q'4,1	Q5,1	Q5,1	Q6,1	Q6,1	r	r
	LSB	Qo,0	Q'o,0	Q1,0	Q'1,0	Q2,0	Q'2,0	Q3,0	Q3,0	Q4,0	Q'4,0	Q5,0	Q5,0	Q6,0	Q6,0	r	r

7.7 RT (Retransmitter)

The Retransmitter block connects the Protocol Encoders and the Retransmit path from the CPRI Input Converters to the transmit path of the Antenna Interface IP block.

The Retransmitter can operate in the following configurations.

A received link can be looped back out the transmit port (retransmission). This is useful for WCDMA UL Daisy Chain Broadcast, or for a direct loop-back for debug and test.

A created “link” may be sent out the transmit port. This may be used for the start of a DL daisy chain.

The Aggregator functionality performs either insertion or addition for DL daisy chain.

- Insertion is used for
  - OBSAI packet message insertion
  - OFDM DL where each node may insert an entire DL stream
  - WCDMA DL
  - First node in the daisy chain
  - In the event device were to be used for UL
- Addition is used for
  - WCDMA DL daisy chain aggregation

The Retransmitter function is positional based as opposed to address based. Therefore, the positional format of a received stream must be known in order to properly align with the created stream from the Protocol Encoder. The ingress position must exactly correspond with Transmission rules programmed into the PE. This is not required for retransmission of received data.

When used for message insertion, the Retransmitter can overwrite an empty message, when the retransmit message slot is not used, or non-empty message, when the message slot is used upstream in a daisy chain but not used downstream, with a new message. This is useful in creating packet data messages, or insertion of uplink or downlink streams.

When the Retransmitter is programmed to use both the ingress received link data and PE data, the PE controls the aggregation of the RT. Based on transmission rules, signals provided by the Protocol Encoder may modify the reTransmitter functionality (e.g. AxC summation and control message insertion) on a per message slot basis.

The reTransmitter supports addition of 8- or 16-bit IQ data. The Addition function of the reTransmitter sums the payloads of two 16-bit I and two 16-bit Q DL data messages together or two times two 8-bit I and two 8-bit Q UL messages together. If the result of summing would be to overflow and sign wrap, saturation logic limits the addition to the max positive or negative values.

In case of OFDM, addition means inserting valid IQ data into dedicated empty slot for each AxC in the chain. It is user’s responsibility to setup transmission rule for each DSP in the chain not to make any data collision. Chapter 9 shows some program example how to setup transmission rule for OFDM.

Saturation is performed on each adder output. Saturation logic detects addition overflow and replaces the overflowed value with a value based on the selected bit width.

The saturation values are:

- 8-bit data: ± 127
- 16-bit data: ± 32,767

Table 7-11. RT Saturation Logic

Number of bits	Sample Size	Input Range	Output Result
8	8	8’h80 – 8’hff (Positive) +CO	8’h7f
8	8	8’hc0 – 8’h81 (Negative) + CO	8’h80
16	16	16’h8000-16’hffff (Positive) + CO	16’h7fff
16	16	16’hc000 – 16’h8001 (Negative) + CO	16’h8000

7.7.1 15-bit Saturation Operation for CPRI (7-bit follows the same rule)

When bit 15:14=='b01, then the saturation output will be 0x3FFF.

Ex) Sent: 0x40004000 => output: 0x3FFF3FFF

When bit 15:14=='b10 and bit 15:0 != 0x8000, then saturation output will be 0x4000.

Ex) Sent: 0x80018001 => output: 0xC000C000 (bit 15 is don't care, so it is same to 0x4000)

When bit 15:0=='0xC000, then saturation output will be 0x4000.

Ex) Sent: 0xC000C000 ? output: 0xC000C000 (bit 15 is don't care, so it is same to 0x4000)

All other cases, there is no change.

The MSB (Bit 15) is used to differentiate positive and negative signal but it is don't care bit for the output.

7.7.2 16-bit Saturation Operation (8-bit follows the same rule)

The 'Negative' saturation will occur if the RT receives a 16 bit aggregation command from the PE. If PE **pe_dma0chan** register **rt_ctl** field option is **ADD16**, the negative saturation mechanism will work. The aggregation results in a 17 bit value with carry equal to 0x18000 or bits 16:15 are equal to 'b10.

7.8 PD (Protocol Decoder)

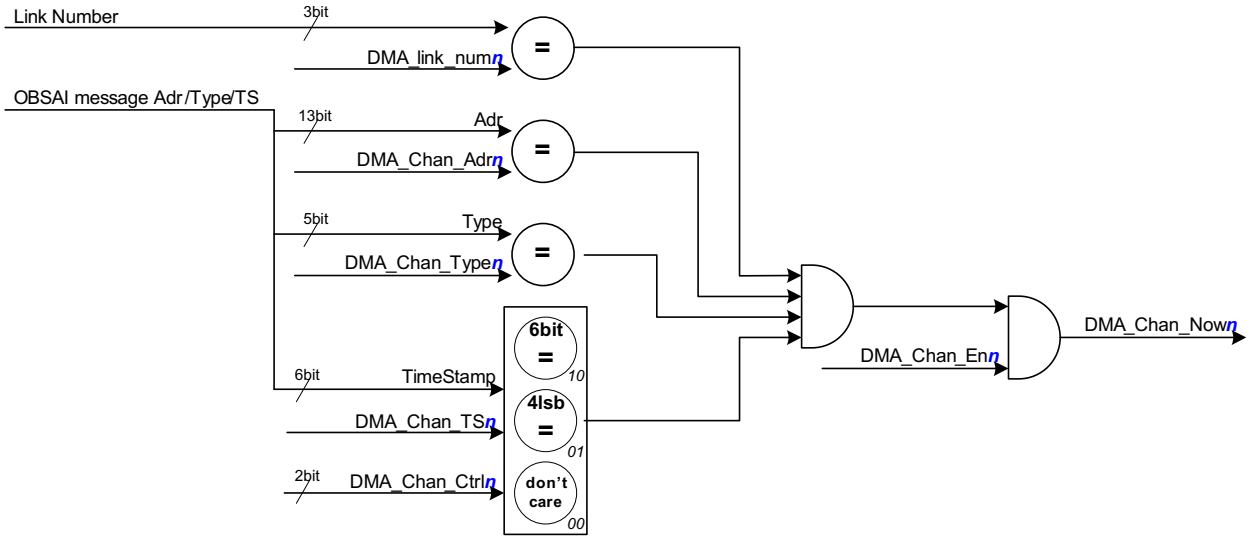
The PD front end is link based, with six different copies of the identical circuit for purposes of dedicated hardware per link. The PD back end is a single circuit that is TDM shared between the different links.

Internally, PD has three modules inside. Route and Pack module has six identical copies and Stage module has TDM interface to DB module. Below sections describe each internal module in PD and how to configure registers.

7.8.1 PD Route Module

7.8.1.1 OBSAI Route and DB buffer Channel Number

Figure 7-17. OBSAI DMA Index Circuit (Header-Based Reception)



For each of 128 possible DB buffer Channels, the OBSAI Route circuit is a set of 128 parallel compare circuits. When a compare circuit fires, that particular DB buffer Channel associated with that compare circuit yields an OBSAI message. Each of the compare circuits has a one-to-one mapping to the DB buffer Channel number.

In the event that a user would like to map multiple OBSAI addresses to a single Destination, the user would accomplish this by mapping multiple DB buffer Channels to a single Multicore Navigator Destination queue.

PD_DMACHAN[128].CHAN_EN: (OBSAI) used to qualify each individual compare circuit. Channels that are Off will not inject traffic into the rest of the PD circuits.

PD_ROUTE[128].ROUTE_LK: Link Number for which this circuit is to be allocated. Allows for same OBSAI address to be reused on an adjacent link

PD_ROUTE[128].ROUTE_TYPE: OBSAI Type to compare

PD_ROUTE[128].ROUTE_ADR: OBSAI Address to compare. user may define max 13 bit unique address for all DSPs in the whole system

PD_ROUTE[128].ROUTE_TS: In modes that extend TS for addressing, this field indicates the TS value to be compared. This field is mainly used for OBSAI control packet or Generic packet mode.

PD_ROUTE[128].ROUTE_MASK: This indicates how many bits of the TS should be compared

7.8.1.2 CPRI Route (DBM) and DB Buffer Channel Number

Starting at the beginning of the CPRI Phy frame, PD is simply counting through the CPRI framing to find the CW vs. AxC region. Each of these regions is handled in a different way.

7.8.1.2.1 CPRI CW

The PD simply enumerates the CPRI CW within a hyper-frame as 0-255. The PD uses the **[6]PD_CW_LUT[256]** to indicate how to use these CPRI Control Words. Some of the CW has fixed use, but most are available to pass generic user data.

[6]PD_CW_LUT[256].CW_EN: 0x1 indicates CW should be considered for packet extraction, where 0x0 indicates to ignore CW for packet extraction.

[6]PD_CW_LUT[256].CW_CHAN: Indicates 0-3 possible unpacking CW packet packing circuits. Allows for four separate streams of CW traffic

AIF2 allows packet traffic to be passed over otherwise unused CPRI AxC bandwidth. For all CPRI packet traffic, there are only the four packet packing circuits. It is the responsibility of the user not to dual allocate these resources for both CW and AxC packet traffic.

7.8.1.2.2 CPRI AxC

AIF2 uses the OBSAI Dual Bit Map FSM (DBMF) concept for configuring the use of CPRI bandwidth between AxC. The DBMF is essentially a simple round robin TDM of AxC with the addition of a programmable bubble insertion at the end of each cycle of round robin. For CPRI, the Modulo rule is not used at all; instead, the user can set up a maximum of two DBMs (One for channel 0 ~ 63 and the other one for channel 64 ~ 127) per link.

The CPRI DBMF interleave operation is done on a sample-by-sample basis. A sample is considered to be

- 8-bit or 16-bit mode: 32 PHY AxC bits
- 7-bit mode: 28 PHY AxC bits
- 15-bit mode: 30 PHY AxC bits

[6]PD_DBM.DBM_X: For WCDMA, it is CPRI Max number of AxC that fit into a link. For LTE, it is max number of LTE symbol. For TD-SCDMA, it is max CPRI sample (could be 16 bit or15 bit) numbers in three basic frames. Programmed value is (X-1)

[6]PD_DBM.DBM_XBUBBLE: In the event of adding bubbles for rate matching, indicates how many bubble to insert (more programmability than permitted in the OBSAI DBMF). Bubble size is same to CPRI one sample size. 0x0: one bubble, 0x1: two bubbles…

[6]PD_DBM_1MAP[4].DBM_1MAP[32]: Starting with reg[0],bit[0] indicates whether a bubble burst should be inserted at the end of a burst of AxC samples (Burst of AxC is a burst consisting of one sample for each of DMB_X AxC). Bitmap usage increments by one position after each burst of DBM_X samples.

[6]PD_DBM.DBM_1MULT: How many times DBM_1MAP should be repeated prior to performing DBM_2MAP

[6]PD_DBM.DBM_1SIZE: Number of bits of bit DBM_1MAP to use.

[6]PD_DBM_2MAP[3].DBM_2MAP[32]: Same to DBM_1MAP

[6]PD_DBM.DBM_2SIZE: Number of bits of bit DBM_2MAP to use

The DBMF algorithm increments AxC channel index through the bit maps, one bit per burst of AxC samples. If the bit is 0x1 then a burst of bubbles (zeros) is inserted before the next burst of AxC samples. Bit map1 will repeat several programmable times, followed by one sequence of bit map 2. The used length of map1 and map2 is programmable. In the event that map2 is programmed to a length of 0, then map2 is unused.

[6]PD_CPRI_ID_LUT[128].CPRI_DMACHAN: LUT mapping the CPRI DBMF X count to DB buffer Channels. LUT is indexed locally by X Count (0 ~ 127). Max X is 128, so it is corresponded to max 128 channels or 128 samples. User programs DB buffer Channel corresponding to each enabled X count.

[6]PD_CPRI_ID_LUT[128].CPRI_X_EN: 0x1 enables X slot to be carrying valid traffic to be packed and transported to DMA circuits.

[6]PD_CPRI_ID_LUT[128].CPRI_PKT_EN: Indicates that a CPRI AxC should carry packet data and not AxC data. Two LSBs of [6]PD_CPRI_ID_LUT[128].CPRI_DMACHAN indicates which of four DB buffer Channels. 4B/5B encoding is required.

[6]PD_CPRI_ID-LUT[128].CPRI_8WD_OFFSET: This Indicates fine AxC offset for PD to select which word can be transferred to DB FIFO first. Used in the front end of PD to align word data into Qwords. The unit of this offset is word(4 bytes). PD will choose which word is the first valid data that could be transferred to DB FIFO. Bit[1:0] are offset into a QWord. Bit[2] gives RSA double QWord alignment.

7.8.2 PD Pack Module

7.8.2.1 PD Pack CPRI Packet Delimitation

7.8.2.1.1 CPRI

The first stage of the packing operation is looking for CPRI packet boundaries. There are three different packet delimitation options:

- 4B5B encoding: uses 20% of the bandwidth (five-bits representing each four bits of possible data) for purposes of creating SOP and EOP indication in the data stream
- Hyper-frame Delimitation: Packet boundaries are assumed to be on Hyper-frame boundaries.
- Null Delimiter is a byte-by-byte operation where a single byte is reserved to indicate null traffic (the absents of traffic). Each byte is represented by 9 bits where the MSB indicates a K-char byte.

[6]PD_LINK_B.CW_PKT_DELIM: For CPRI Control Words, Indicates whether 4B/5B or Null Delimiter should be used in packet delimitation. Also indicates if no packet data should be extracted from CPRI CW.

[6]PD_LINK_B.CPRI_AXC_PACK: Indicates how many significant bits are sent over CPRI AxC. This info is used for packing received AxC bits for purposes of sending/receiving packet data over CPRI AxC slots.

[6]PD_LINK_B.CPRI_NULLDELM: 9 bit field indicating which pattern of 9 bit data should be used as the Null Delimiter. Bit[8] is used to indicate K-char. If this bit is zero and that means this null delimiter is not K char, but in this case, user can not use this 8bit delimiter as a data because some data might have same value as delimiter. So User can use unused K char like K27.7 (0xFB) for this purpose. In this case, Bit[8] should be set as 1. Well known K char for AIF like K28.5 or K28.7 can not be used for null delimiter.

Both the Null Delimiter and 4B/5B decode circuits strip delimiters prior to the next operation. Internally, the SOP and EOP information is passed with sideband-signals, so no information is lost.

In cases of both Null Delimited and 4B/5B encoded packets, the minimum length packet that AIF2 supports is four bytes of payload:

- **4B/5B**: by definition requires 10 bits of SOP and 10 bits of EOP that are not part of the packet payload. The minimum payload is four bytes (actually 40 bits in 10B representation.
- **Null Delimiter**: There is a minimum of one Null Delimiter between packets. The first (byte that is not a null byte) is both SOP and the first byte of payload... The last byte (just before the first Null character) is both EOP and the last byte of payload.

For hyper frame delimited packets, the minimum packet size packet is equal to a single control word. This is two bytes for a 2x rate link.

7.8.2.2 PD Pack CPRI Null Delimitation

The Null delimitation circuit receives up to a 32-bit input bus and has an output 32-bit output bus. The possible input cases are:

- AxC (all four byte lanes active)
  - 32-bit
  - 28-bit
  - 30-bit
- CW (byte lane enables identify valid bytes.
  - 2xmode: 2 byte
  - 4xmode: 4 byte
  - 5xmode: 4byte/1byte interleaved
  - 8xmode: 4byte

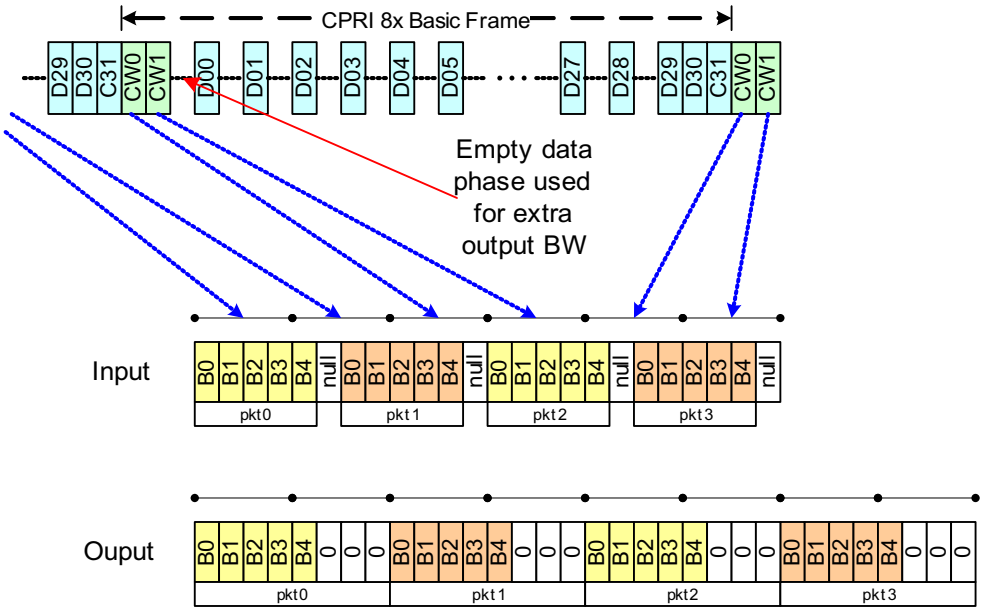
Packets:

- Minimum 4 bytes
- Minimum one Null delimiter between packets
- AxC packets must be nx4 bytes in length
- CW packets can be any length of four bytes or more

In any given input data phase has the potential to span two sequential packets. The basic functionality of the Null delimiter circuit is to identify packet SOP and EOP and partition the outputs such that any single output phase belongs to only one packet. The Null Delimiter circuit uses appropriate zero padding in order to accomplish this functionality.

This internally defined Null Delimitation process has the unique issue of potentially requiring greater output bandwidth than input bandwidth. A very good example of this is minimally spaced five-byte packets. Generalizing this issue, this issue occurs for packet of size (nx4 +1) or (nx4 + 2).

Figure 7-18. PD_Pack_CPRI_Null_Delim Corner Case



The example above shows the output of the CI submodule for the 8x rate CPRI mode. In the example, all control words are mapped to a single packet channel — a five-byte packet with a single null delimiter between packets. On average, for every three input data phases the null delimiter circuit must output four data phases.

There is a guaranteed empty input data phase after CW1 in 8x mode. This extra data phase is used by the Null Delimiter circuit to output “extra” bandwidth due to the unpack operation.

Packet data which is passed in CPRI payload (previously used for AxC data only) must always be (nx4) bytes in length. With this restriction, the output bandwidth expansion problem does not exist, however packets in CPRI payload do have the extra complexity of potentially being packed into 7-bit or 15-bit modes. In these modes, the Null Delimiter (and 4B/5B delimiter) circuits need extract the useful bits, consume whole (byte) segments and store any residual bits to concatenate with the next input data phase.

7.8.2.3 PD Pack OBSAI Packet Delimitation

OBSAI AxC traffic simply passes through these first stages. OBSAI packet data utilizes the OBSAI header information. SOP and EOP are extracted from this information and SOP/EOP indication added to payload data, passed through block.

7.8.2.3.1 Ethernet Timestamp

- SOP: 6’b100000
- MOP: 6’b000000
- EOP: 6’b1XXXXX:XXXXX cannot equal 0 and indicates the number of bytes from the start of RP3 payload containing MAC frame data (counting started from the byte after the header) This XXXXX value is passed to the PD_Stage for purposes of calculating packet length.

Ethernet uses the MSB of TS to indicate SOP or EOP. For EOP, the five LSB can never equal 5’b1_1111. Using this fact, PD can deterministically discriminate SOP from EOP. MOP TS is always equal to 6’b00_0000.

Unlike other OBSAI Types, Ethernet has the unique characteristic that its length is not necessarily an even number of Qwords. The 5LSB are used to indicate the number of valid bytes in the last (EOP) OBSAI message.

7.8.2.3.2 Generic Packet Timestamp

- SOP: 6'b10XXXX
- MOP: 6'b00XXXX
- EOP: 6'b11XXXX (XXXX: is an extension of OBSAI address and is the same for all elements within the packet) This XXXX value is unused and discarded.

7.8.2.3.3 Control Message Timestamp

- Generic Control Message:
  - Timestamp = 6'b000000
- Control Message for Air Interface Synchronized Operations:
  - Timestamp = 6'b000001

[6]PD_TYPE_LUT[32].OBSAI_PKT_EN: look up using five-bit(32 cases and this number is matched with LUT array number) OBSAI Type (from OBSAI msg header) indicating that the message is part of a packet and not antenna streaming data

7.8.2.4 PD Pack 4 Channel Map

CPRI control word handling (and CPRI 4B/5B packet data in AxC) is a special case so, channel mapping does not fit into the AxC LUT of ID_LUT from the previous stage. The four possible streams of CPRI data are given their own DB buffer Channel mapping registers. It is the responsibility of the user not to allocated a DB buffer Channel to more than one PHY stream.

[6]PD_PACK_MAP.PACK0_DMA_CH: Maps the CPRI packet stream 0-3 to DB buffer Channel 0-127

[6]PD_PACK_MAP.PACK0_EN: 0x1 enables the channel

…….

[6]PD_PACK_MAP.PACK3_DMA_CH: Same to pack0

[6]PD_PACK_MAP.PACK3_EN: Same to pack0

7.8.2.5 PD Pack Ethernet Header Strip

The Ethernet circuit simply strips the Ethernet preamble and start of frame. The Ethernet header is always eight bytes, which is very convenient because AIF2 has an internal data format of four bytes. So, two data phases strip very nicely. When enabled to strip, the stripping circuit blindly strips without checking that the bytes indeed are the correct Ethernet Preamble and start of frame.

- Preamble: Constant value: 8'b1010_1010
  - Ingress: Stripped off by AIF2
  - Egress: Added by AIF2
- Start of Frame: Constant value: 8'b1010_1011
  - Ingress: Stripped off by AIF2
  - Egress: Added by AIF2

7.8.2.5.1 CPRI

There are four possible channels of Ethernet traffic. Each of four channels has a separate programmable MMR bit that directs the stripping circuit to strip the first eight bytes.

SOP always restarts the stripping operation. In the event of a partial packet, the operation will always start clean at the beginning of a packet.

[6]PD_LINK_B.ETHNET_STRIP: CPRI bits 3-0 enable (active high) the stripping of Ethernet preamble and SOF for each of four possible streams of CPRI Ethernet traffic.

7.8.2.5.2 OBSAI

The OBSAI header indicates whether or not a steam is Ethernet or some other type. PD allows for any of 128 DB buffer Channels of OBSAI to contain Ethernet traffic. The enable bit of Ethernet striping control applies globally to all OBSAI channels per link.

[6] PD_TYPE_LUT[32]. ENET_STRIP: OBASAI Type lookup, indicates whether to strip 1st eight bytes of packet.

[6]PD_LINK_B.ETHNET_STRIP: bit 0 enable (active high) the stripping of Ethernet headers for all of 128 possible channels. OBSAI Header type indicates Ethernet type. Bits 1-3 unused.

7.8.2.6 PD CRC

The CRC circuit performs CRC check on incoming PD packets. The internal interface to the CRC circuit is a quad-byte interface. The delimitation circuits will pack and LSB align data arriving into the CRC circuit, but the last data phase may not be completely full word, supplying {1, 2, 3, 4} bytes of data. The EOP indicates the last valid byte. The CRC circuit supports up to 128 streams and maintains an intermediate state RAM in order to support context switching between the streams. SOP always restarts the CRC calculation and EOP triggers the circuit to generate the CRC_bad signal

8, 16, and 32-bit CRC polynomials are supported. The 16-bit and 32-bit polynomials are fixed and use the Ethernet and OBSAI standard polynomials. The 32-bit Ethernet polynomial is $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$ and has the CRC value complemented in the received packet. The 16-bit OBSAI polynomial is $X^{16} + X^{12} + X^{5} + 1$ and results in a 0 when the CRC is run on the entire packet including the received CRC.

The eight-bit polynomial is programmable via MMRs. Only one eight-bit polynomial is supported per link. The CRC polynomial and the initial value are programmable. No complementing of the CRC is supported in the generated packet so the compare value will be against a 0.

[6]PD_LINK_B.CRC_8POLY: Programmable polynomial for CRC8

[6]PD_LINK_B.CRC_8SEED: Programmable seed value for CRC8

7.8.2.6.1 CPRI

Only four streams support CRC check; these are the four supported CPRI packet streams. In CPRI mode, these four streams are explicitly controlled via MMR bits to enable CRC check and configuration of the CRC width to use (only positions 0-3 are used in the context RAM).

[6]PD_CPRI_CRC.CRC0_TYPE: CPRI stream 0: selection of CRC width

[6]PD_CPRI_CRC.CRC0_EN: CPRI stream 0: enable of CRC checking
……

7.8.2.6.2 OBSAI

In OBSAI mode, the CRC operation is controlled by the Type field in the OBSAI header (of each OBSAI message). The received Type field is used to index the Type LUT that indicates whether or not a CRC is checked and indicates the CRC width. The lookup is performed outside the CRC circuit an passed into it as control signals. In OBSAI mode, all streams are permitted to carry a CRC, so the CRC circuit supports a context RAM location for each possible stream.

[6]PD_TYPE_LUT[32].CRC_TYPE: CRC width programmed by OBSAI Type.

[6]PD_TYPE_LUT[32].CRC_EN: CRC check enable by OBSAI Type

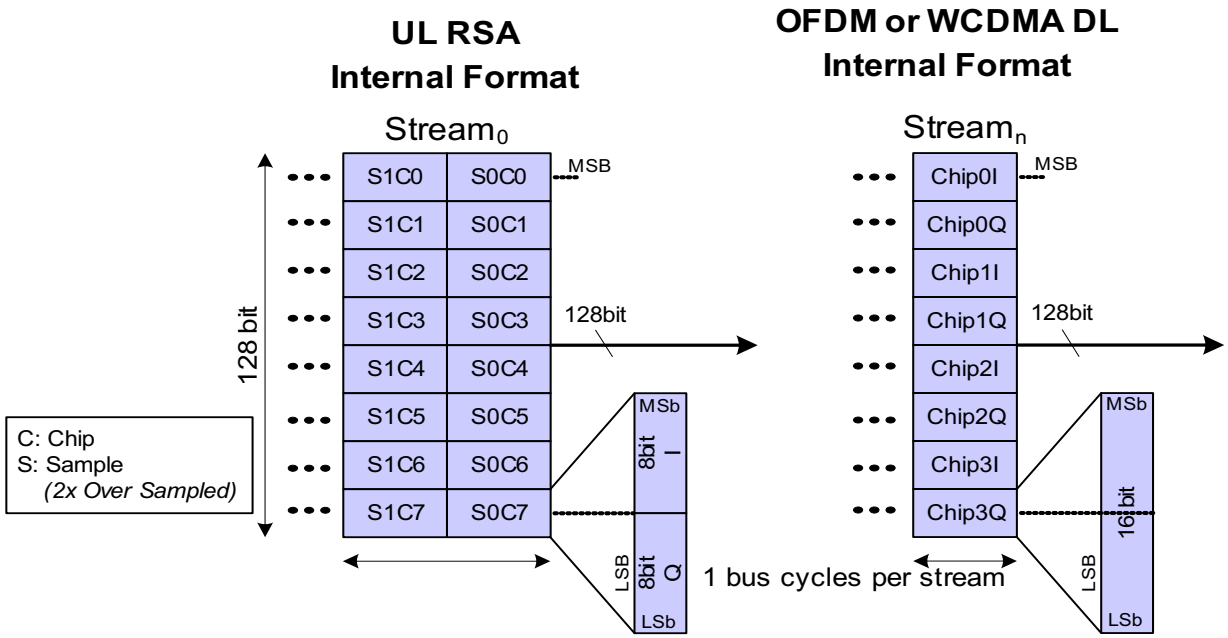
7.8.3 PD Stage Module

7.8.3.1 PD Stage

7.8.3.1.1 FSM

Data is received as quad-bytes and packed into quad-words. The FSM counts through the data phases in 4's or 8's depending on the data format. The two LSB of the count is used to control the barrel shift. For AxC in RSA Mode, an eight count triggers the FSM to post an entry for consumption while in other modes a four counts or EOP triggers the FSM to post an entry for consumption.

Figure 7-19. PE/PD to DB internal Data Format



7.8.3.1.2 RSA Convert (WCDMA UL)

AIF2 internal format (called RSA format) for WCDMA UL is to arrange even samples and Odd sample together into sequential quad-words. Over the PHY, these samples arrive interleaved. The RSA Converter is a deinterleaver that spans two address locations from the Stage RAM.

PD_DMACHAN_B[128].DATA_FORMAT: When the data format is RSA type, both the staging RAM and RSA Convert circuit use a double data phase for processing traffic. The RSA converter deinterleaves the PHY traffic into odd and even samples.

7.8.3.2 PD FRAME

A framing counter is maintained for each of 128 possible streams (when in AxC mode). The configuration of the frame counter is MMR configurable. The start of the timers is dependent on the timing of each AxC.

7.8.3.2.1 OBSAI

AT timer value is passed along with each OBSAI Q-Word which is the AT timer value when the OBSAI message is received. The AT timer value is used as a comparison against a programmed window. The programmed window is created by a center PD_DMACHAN_A[128].AXC_OFFSET and a plus/minus offset adder [6]PD_LINK_A.AXCOFFSET_WIN.

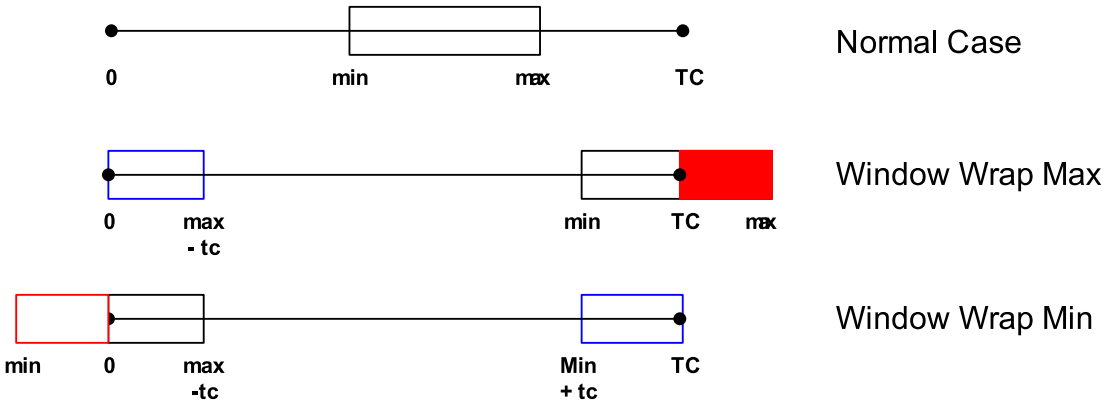
PD_DMACHAN_A[128].AXC_OFFSET: Center of the timing window for comparison with OBSAI message received time. Programmed in 307.2MHz Clocks, relative to the RadT Frame Boundary

[6]PD_LINK_A.AXCOFFSET_WIN: Plus or minus offsets for window creation (for comparison with OBSAI message received time.)

For unrecoverable OBSAI TS failure, the TS checker will fail the channel until the next Radio frame boundary (or in case of GSM, until the next GSM slot). Failing the channel means that all traffic for that channel is dropped on the floor until the next starting opportunity. The appropriate PD_CHAN_STS bit is cleared until the channel is again activated

(WCDMA and Packet) The channel On/Off control is not so precisely controlled. For packet types and WCMDA traffic, the first good data turns the enabled channel on. For packet traffic, this first good enable would be an SOP. For WCDMA AxC traffic, the first good data is any valid OBSAI message for a given AxC.

Figure 7-20. Window Wrap Cases



Normally one thinks of the window centered in the region. In cases were the AxC offset is either near zero or near the RadT_TC, the window can span beyond the extremes of the RadT count. In these cases, the PD circuits adjust to window in order to account for wrap conditions. This operation should be transparent to the user.

7.8.3.2.2 CPRI

The PHY frame boundary is used as a timing reference point. PD will start AxC extraction after AXC_OFFSET number of four samples (quad word size) has passed. After start, counters will simply count without re-synchronization.

PD_DMACHAN_A[128].AXC_OFFSET: Exact offset of an AxC. It is programmed in number of four samples (QW), relative to the CPRI Link (PHY) Frame Boundary. This offset is only applied to OFDM radio standard but not used for WCDMA. (RAC will control external delay for WCDMA)

PD_GLOBAL_EN.SET: Write-only function globally enabling PD, allowing each channel to be individually enabled.

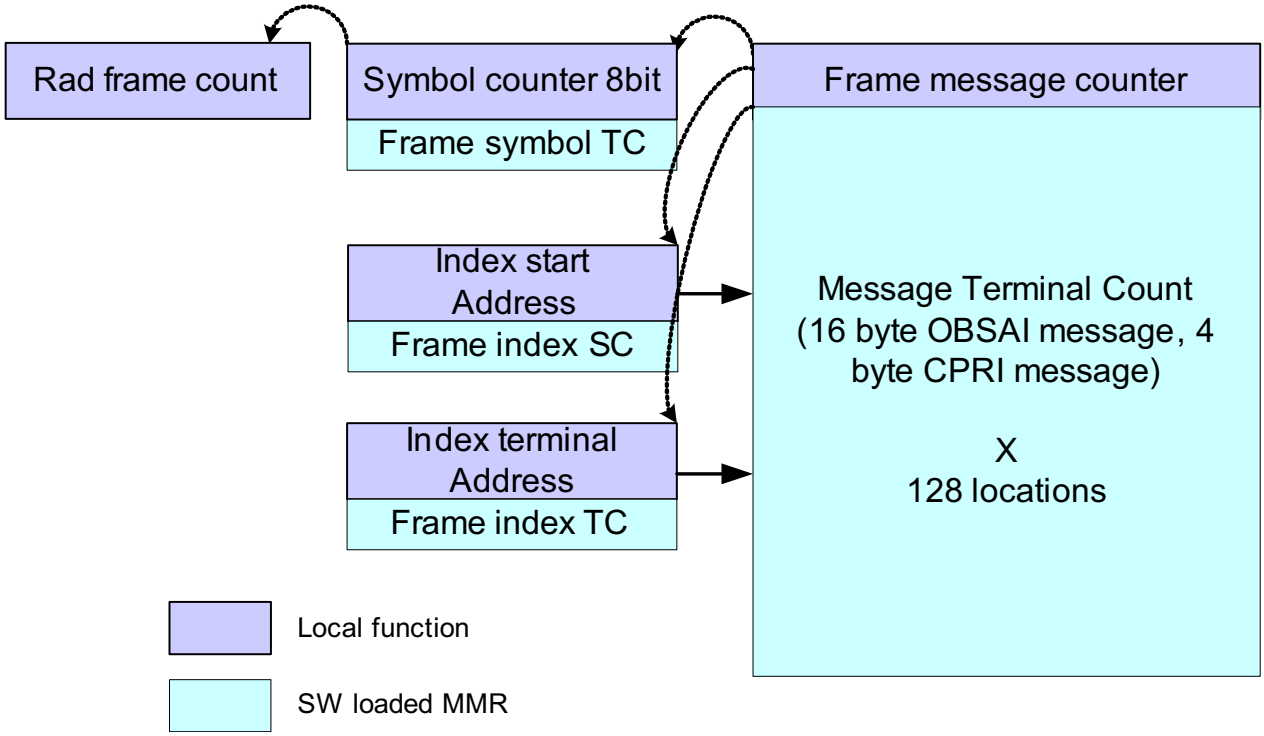
PD_GLOBAL_EN.CLEAR: Write-only function globally disabling PD…

PD_GLOBAL_EN_STS.ENABLE: Read-only status of global enable state.

The global enable is AND'd with each individual channel enable so that both the global enable and individual channel enable must be 0x1 in order for the channel to remain in the "ON" state. After reset condition, the software must write the global enable bit before any channel can activate.

7.8.3.2.3 AxC Frame Counter

Figure 7-21. PD Frame : Framing Counter Circuit



The PD, PE framing counter configuration is radio standard specific and used for {LTE, WiMax, TD-SCDMA, GSM} but at least one symbol TC and frame message TC also should be set for WCDMA. This is very complex in order to satisfy all the special requirements of all the different radio standards. (Please see AT Module section) for a basic understanding of the Frame Timer concept.

There are three main differences:

- PD/PE maintains timer count values independently for each of 128 channels
  - AT maintains 3 {RadT, RadT_UL, RadT_DL}
- PD/PE timers increment based on samples received
  - AT: counts clock cycles
  - PD:
    - OBSAI: count groups of 16 bytes (that is, four samples)
    - CPRI: count groups of 16 bytes (that is, 4x 4 byte samples)
- PD/PE (LTE requirement) simultaneously supports six different sets of terminal counts.
  - AT: supports only one set of terminal counts

In support of the six different sets of LTE timing, most terminal count registers have six different versions. **PD_DMACHAN_B[128].FRM_GRP** programs on an AxC-by-AxC basis which of the six versions to use. The one exception of this register duplication is the **PD_FRM_MSG_TC[128].FRM_MSG_TC,** which is so large that it is used to support all six versions in its 128 deep buffer. LTE only requires six or seven FRM_MSG_TC per version of the LTE counters; the user partitions the 128 deep buffers by dictating a start count (**FRM_INDEX_SC**) and a terminal count (**FRM_INDEX_TC**) for each of the six LTE possible counters.

The FRM_INDEX is a counter that parallels the FRM_SYM count, but programmed to wrap one or several times prior to the FRM_SYM wrap. Used specifically in LTE as a six or seven counts. This FRM_INDEX concept is useful in limiting the supported range of **PD_FRM_MSG_TC[128].**

PD_DMACHAN_B[128].FRM_GRP: Indicates which of six sets of terminal counts to use for each channel.

PD_FRM_MSG_TC[128].FRM_MSG_TC: Number of messages (16 bytes) that are in a symbol/slot for OBSAI. For CPRI, it is the number of 16 byte (4x 4bytes) sample. 128 entries support 128 symbols of different length. 128 regions can be separated into six different regions supporting the six different sets of timer terminal counts.

PD_FRM_TC[6].FRM_SYM_TC: (six sets) Terminal Counter: Number of symbols to count, programmed as a terminal count table.

PD_FRM_TC[6].FRM_INDEX_TC: (six sets) Index parallels FRM_SYM_TC and used for indexing the message terminal count table.

PD_FRM_TC[6].FRM_INDEX_SC: (six sets) Index used for indexing the message start count table.

7.8.3.2.4 Packet State

For each of 128 possible channels, a state bit is maintained indicating whether the channel is “in packet” or “out of packet”. This state information is used in global shutdown to indicate when it is safe to ack clock stop request. The state is also used to determine when a packet channel may transition into an OFF state.

Pkt Mode, In/Out packet is clear. When a SOP is received, the channel is IN_PKT. Once the EOP is received, the channel is OUT_PKT.

In AxC Modes (all radio standards other than WCDMA) the OFDM Symbol or GSM timeslot is considered packet boundaries and determines the IN/OUT_PKT state.

PD_CHAN_STS[4]: Status bit per channel indicating if the channel is ON or OFF

PD_PKT_STS[4].CHAN_PKT[32]: Status bit per channel indicating if the channel is IN/OUT_PKT. SOP or any PKTDMA beginning of packet sets bit. EOP or any closure of Multicore Navigator packet clears the bit. (For WCDMA, this bit is unused and set to 0x0)

7.8.3.2.5 Timestamp Check

For WCDMA, the Timestamp (TS) is simplified only checking that the 4 LSBs of the timestamp have incremented from the last message received.

PD_DMACHAN_B[128].GSM_UL: Chooses between two slight variants of GSM Timestamp protocols.(Basically needed because of a short coming in the OBSAI standard. Normally information such as this is built into the header info)

7.8.3.2.6 Watchdog

The watchdog function is a very course timing function designed to catch missing EOP for GSM traffic. The function is generalized to any possible AxC packet (Multicore Navigator) traffic data. The intent is to detect only a missing EOP for a GSM slot that is very late. “Very late” is relative and programmed by the user. Channels timings are polled in null cycles, so WDOG activity has very course timing.

PD_DMACHAN_B[128].TS_WDOG_EN: Channel by channel enable of watchdog checking. In the event that an EOP is expect, and the TS_WDOG_CNT is exceeded, PD will automatically insert an zero Qword and EOP.

PD_DMA[1].TS_WDOG_CNT: Terminal Count value of the watch dog counter (approximately three sequential terminal counts result in a failure)

PD_DMA[1].WDOG_EOP_ADD: Controls circuit to add an Qword if (and only if) an EOP was expected.

PD_DMA[1].WDOG_EE_CTRL: Controls EE reporting of WDog failures only when an EOP was expected, or in all WDog failures.

The Watch Dog counter simply starts counting once hardware reset is released. The counter starts at zero and counts up to the terminal count indicated by TS_WDOG_CNT, then rolls back over to zero. The Watch Dog state is a four count, starts at zero and increments every wrap of the Watch Dog Count. Each time a channel is serviced, the Watch Dog state is sampled, decremented by one, and stored.

7.8.3.3 PD DMA_IF

PD_GLOBAL[1].DIO_CPPI: A global bit sets whether antenna traffic will be circular buffered for DIO transport or packetized for Multicore Navigator transport. This is no support for mixing modes.

PD Direct IO

Antenna carrier data can have any random offset which is mainly compensated in the RAC. The important function of the PD/DIO is to align DB bursts such that DB is not reading from the same location being written by PD. In support of this, PD implements an AxC by AxC offset of 4 bits. The user normalizes incoming AxC with this offset.

PD_DMACHAN_B[128].DIO_OFFSET:

Added offset to Circular RAM write addressing. Programmed on AxC-by-AxC basis.

In case of 8 QW(Quad word) buffer, the valid range is between 0 ~ 7

PD DB Scheduling for TDD mode

In support of TDD systems, the PD drops Symbols/Timeslots of data without DMA'ing them into the device. AIF2 links are duplex where the air channel is simplex. Total 144-bit symbol bitmap is supported per channel indicating whether or not that symbol/timeslot is to be DMA'ed or dropped.

PD_DMACHAN_B[128].TDD_EN: LSB is for Symbol/Time_slot 0, with a bit for up to 16 symbols/time_slots. 0x1 indicates enable, 0x0 indicates drop.

PD_DMACHAN_C[128].TDD_EN

PD_DMACHAN_D[128].TDD_EN

PD_DMACHAN_E[128].TDD_EN

PD_DMACHAN_F[128].TDD_EN: 128 additional tdd enable/disable bitmap to cover all symbols in one frame.

7.9 PE (Protocol Encoder)

This chapter describes the transmission rules for OBSAI and CPRI, which is one of the most important parts in PE and divided PE internal modules into six small sections. It is Pack, Timer, Frame, DB interface, Message construction and RT Interface. Each small section also shows how to configure MMRs which are related to each section.

7.9.1 OBSAI Transmission Rules

OBASI supports two forms of Transmission rules (Modulo, Dual Bit Map) and 64 rules total for all links. Each rule supports one Modulo and one DBM

Key different features of the AIF2 implementation are as follows:

- Modulo Rules upper limits
  - max Modulo of 2^{12}
  - max Index of 2^{12}- 1
- There are only 64 Modulo Rules (shared between all six Links)
- There are only 64 Dual Bit Map Rules (shared between all six Links)
- AIF2 intends that WCDMA should use a single Dual Bit Map Rule for all AxC within a Virtual Link.
- Dual Bit Map Rule
  - AIF2 extends some of the Dual Bit Map FSM fields to extend the capabilities of this feature
  - AIF2 allows the "Bubble" output of the Dual Bit Map FSM to be mapped to packet switched traffic
  - Dual Bit Map Rules can map to either CirSw(AxC) or PktSw(Generic, Ethernet, Control)streams and can even support a mixture
  - Max X (AxC number)
    - Normal use, may not exceed 64

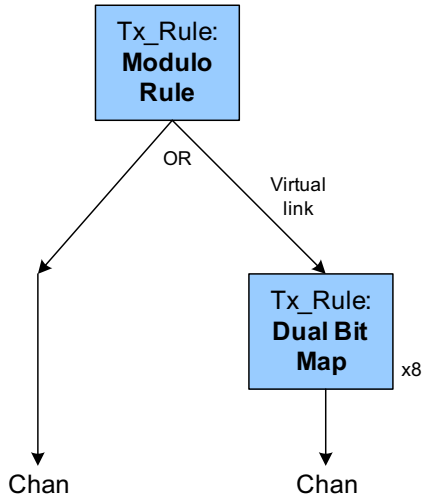
- May not exceed 63 if bubble bandwidth is being used.
- Paired, two bitmaps can be concatenated to form a 128 LUT, consumes two adjacent rules (starting at an even indexed rule)

- AIF2 allows for multiple Rules to map to a single channel (each channel represent a AxC or control stream)

The AIF2 PE Transmission Rule implementation is three parts:

- Modulo Rules
- Dual Bit Map Rules
- Channel Look Up Table

Figure 7-22. PE: OBSAI Transmission Rules



A key factor in the AIF2 implementation is that Modulo Rule Circuits consume large area and power. As a result, AIF2 supports only 64 modulo rules relying on Dual Bit Map Rules for support of all Circuit Switched traffic (regular streaming channels that is, AxC). The Dual Bit Map Rules allow for whole AxC slots to be allocated for Circuit or Packet Switched data, but does not allow the full flexibility that Modulo Rules allow (that is, sharing an AxC slot between multiple PktSw streams).

AIF2 supports the concept of a single DBM rule to index beyond its 64 RAM LUT locations and into the next adjacent DBM rule channel LUT address space. This feature allows a single DBM rule to have and XCNT of 128. When using this feature, user may use a DBMR number 0, 2, 4... and allow it to utilize the memory space of the next higher DBM rule (1, 3, 5...).

The output of the Transmission Rules circuit is a single channel index and enable which triggers message construction and data fetch for that channel.

7.9.1.1 PE OBSAI Modulo Transmission Rules

The counter simply resets to zero at each Link Frame boundary and increments by +1 whenever the appropriate link and data vs. control message is active. When the counter reaches a terminal count of (Modulo – 1), the counter will wrap back to 0.

For a given link, only one Modulo rule may fire in a given cycle. It is considered a programming error for multiple modulo rules to fire and may yield unpredictable results.

PE_MODTXRULE[64].RULE_MOD: Modulo Terminal count. Terminal count is the real OBSAI Modulo minus 1.this modulo rule is working as a base of the DBMR or independently working like AIF1 modulo rule. If user wants to use bigger modulo than 64, one DSP can use 64 modulo amount of bandwidth and other DSP may use the rest of the total bandwidth.(that is, DSP0 Modulo rule_mod = 127 and use rule_index 0~ 63, DSP1 Modulo rule_mod = 127 and use rule_index 64 ~ 127)

PE_MODTXRULE [64].RULE_EN: Enable or disable the transmission rule

PE_MODTXRULE [64].RULE_CTL_MSG: if this bit is set to one, modulo rule operates for OBSAI control message. It will work for AxC messages in other case. AIF2 transmission rule is totally separated for AxC data slot and Control slot.

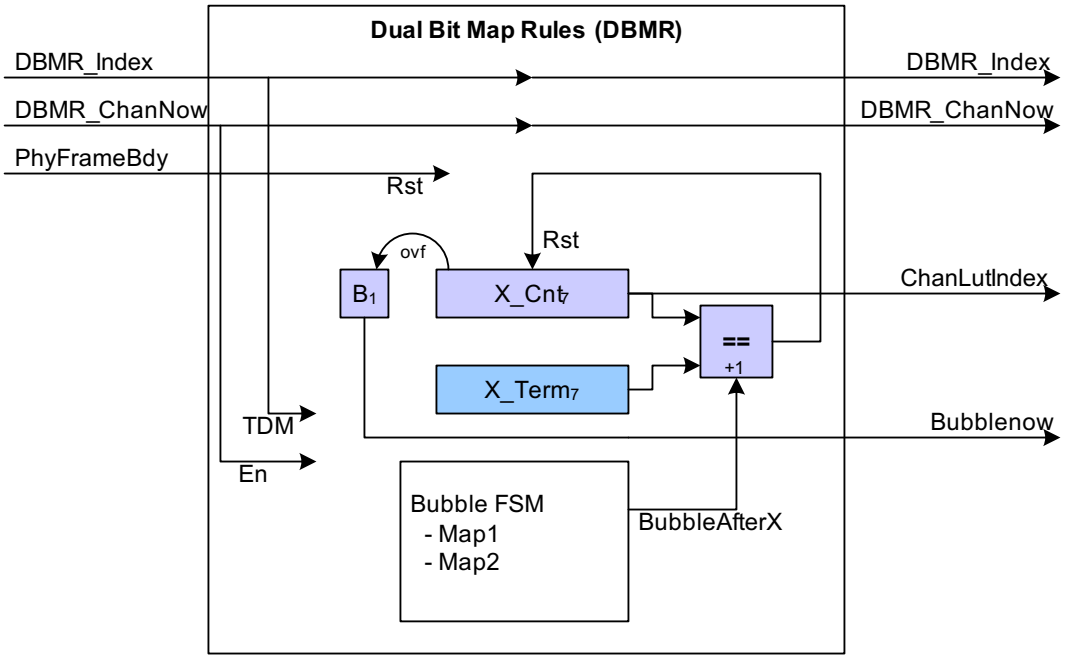
PE_MODTXRULE [64].RULE_INDEX: Transmission Rule Index indicates at which count the rule will fire. If user bypass the DBMR and try to use legacy modulo rule for 16 AxCs, rule_mod value of 16 rules will be 15 and index value of each rule will be 0,1,2,3,…….15

AIF1 easilly allow to move the position of control slot by configuring 84 LUT but AIF2 doesn't have that kind of concept any more and all control slot has fixed position by setting one or multiple of this modulo rules

PE_MODTXRULE [64].RULE_LK: Indicates which link the rule is allocated to. If user wants to change the link, this value also should be changed. DBMR is working on modulo rule, so all DBM rules running on the modulo rule use the same link selected by this field.

7.9.1.2 PE OBSAI Dual Bit Map Tx Rules

Figure 7-23. PE: OBSAI Dual Bit Map Rules



The Dual Bit Map Rules (DBMR) circuit is basically a counter that circularly TDM X channels. Every time the Modulo Rule Fires, the next channel is serviced. The biggest twist, is that at the end of round robin TDM servicing of X channels, a gap of one message is added programmable. The gap is controlled by the "Bit Maps"

Each PHY frame boundary, the Dual Bit Map Rule resets FSMs and counters to the starting state. The Modulo Rule counter holds this reset value for one Modulo Count. The extension of the frame boundary signal is necessary because the DBM Rules are TDM processed and will not all be accessible in the single clock cycle that frame boundary occurs. The DBMR circuit is both TDM of links and rules within a link; the circuit supports FB for one link while other links are at mid-points of FB.

The Bubble FSM consists of a state machine following the Dual Bit Map algorithm. In every state, a single bit of either of the two "Bit Maps" indicates:

- 1'b1: after the "X" count, one additional count prior to rewinding X count
- 1'b0: after the "X" count, simple start over again

The extra "count" is referred to here as a "bubble". During this phase of the count, the output reflects this "bubble" condition (the Chan_Lut operations use this "bubble" slot, allowing a PktSw channel to use this extra bandwidth)

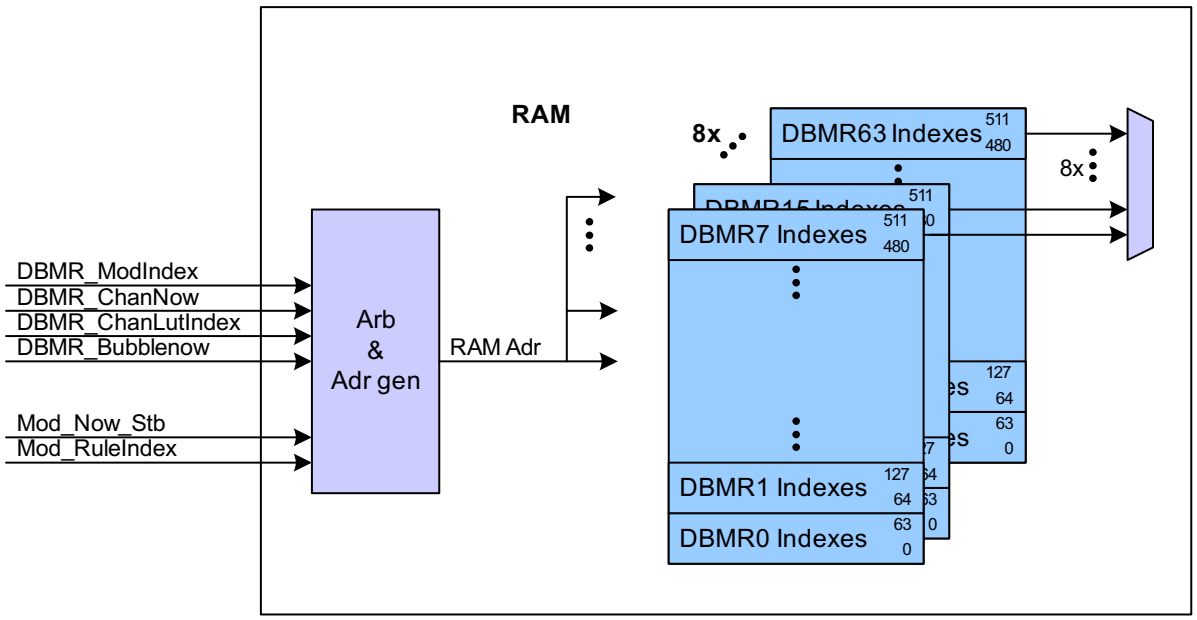
Table 7-12. OBSAI Dual Bit Map FSM Fields

Parameter	Definition	OBSAI Width (bits)	AIF2 Width (bits)
X	Maximum number of AxC or sample that will fit into a given virtual link. Set X-1 value to the register. If X =1, means DBMR bypass mode.(DBMR disabled)	-	6
Bit_Map_1_Mult	Number of times the first bit map is repeated. Set N-1 value	-	8
Bit_Map_1	Value of the first Bit Map. The Bit Map is read starting from the leftmost (MSB) bit.	80	128
Bit_Map_1_Size	Size of the first Bit Map (number of bits). Set N-1 value	7	7
Bit_Map_2	Value of the second Bit Map. The Bit Map is read starting from the leftmost (MSB) bit.	48	96
Bit_Map_2_Size	Size of the second Bit Map (number of bits). Set N value. Zero value means Bit map 2 is not used.	6	7

The algorithm is pretty simple. After every X or X+1 count, the use of the bit map is advanced by one position. Bit_Map_1 is used Bit_Map_1_Mult times then Bit_Map_2 is used once. Then the operation is repeated. When “using” a bit map, only a portion of the bit map is used, indicated by Bit_Map_1_Size and Bit_Map_2_Size.

7.9.1.3 PE Transmission Rules Channel LUT

Figure 7-24. PE: OBSAI Channel Look-Up, Part1

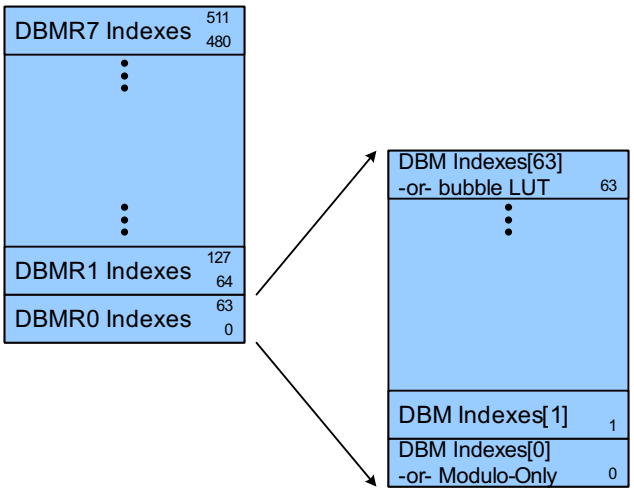


The channel LUT uses mapping transmission rule indexes from the 64 rules into DB channels, is split across eight different RAMs. The partition of this function facilitates the reuse of the LUT for CPRI mode where a LUT is dedicated per link.

CPRI RAM usage:

- Link0: Ram0, 0-127 (address 128-511 unused)
- Link1: Ram1, 0-127
- …
- Link5 Ram5, 0-127
- Ram 6 and 7 are unused in CPRI

Figure 7-25. PE: OBSAI Channel Look-Up, Part 2



For the most part, the tables are a simple translation between the DBM XCNT index to a DB channel. There are a few exceptions:

- Index[63]: In the event that bubbles are to be used as valid bandwidth, the bubble index will always be the 64 th location.
- This does limit the XCNT to 0-62 (loosing one possible XCNT=63)
- Index[0]: In the event that the DBM is disabled (used for single channel with X=0), and the Modulo rule is to be used as a single channel, then location 0 is used as directly as the channel lookup.

CPRI mode does not support the use of bandwidth in the bubbles.

PE_RULE_CHANLUT0~7[512].CHANINDEX: One of 128 DMA channel number, LUT maps rule indexes to DMA channels.

PE_RULE_CHANLUT0~7[512].CHANINDEX_EN: Enable or disable the channel rule

PE_RULE_CHANLUT0~5[512].CPRI_PKT_EN: Dictates the cpri payload is to be used as AxC (normal) or Packet traffic. Channel LUT 6 and 7 doesn't have this field because each LUT(0 ~ 5) is matched with each link.

[6]PE_LINK[1].OBSAI_BUB_BW: OBSAI Only. If this bit is set to 1, DBM rule Bubbles are used for traffic. Channel is indicated by 64th location of Channel Rule LUT. Recommended for Pkt traffic only and recommended only for cases where XCNT is less than 64

7.9.2 CPRI Transmission Rule

AIF2 uses the OBSAI Dual Bit Map FSM (DBMF) concept for configuring the use of CPRI bandwidth between AxC. The DBMF is essentially a simple round robin TDM of AxC with the addition of a programmable bubble insertion at the end of each cycle of round robin.

[6] PE_CPRI_DBM.DBM_X: For WCDMA, it is CPRI Max number of AxC that fit into a link. For LTE, it is max number of LTE symbol. For TD-SCDMA, it is max CPRI sample (could be 16 bit or15 bit) numbers in three basic frames. Programmed value is (X-1).

[6] PE_CPRI_DBM.DBM_XBUBBLE: In the event of adding bubbles for rate matching, indicates how many bubble to insert. Bubble size is same to CPRI one sample size. 0x0: one bubble, 0x1: two bubbles 0x2: three bubbles 0x3: four bubbles

[6] PE_CPRIDBM_1MAP[4]: Starting with reg[0], bit[0] indicates whether a bubble burst should be inserted at the end of a burst of AxC samples (Burst of AxC is a burst consisting of one sample for each of DMB_X AxC). Bit map usage increments by one position after each burst of DBM_X samples.

[6] PE_CPRI_DBM.DBM_1MULT: How many times DBM_1MAP should be repeated prior to performing DBM_2MAP. Programmed value is N-1.

[6] PE_CPRI_DBM.DBM_1SIZE: Number of bits to use 1-100 from DBM_1MAP Programmed value is N-1.

[6] PE_CPRIDBM_2MAP[3]: Same to DBM_1MAP.

[6] PE_CPRI_DBM.DBM_2SIZE: Number of bits to use 0-70 from DBM_2MAP. Programmed value is N.

The DBMF algorithm increments through the bit maps, one bit per burst of AxC samples. If the bit is 0x1 then a burst of bubbles (zeros) is inserted before the next burst of AxC samples. Bit map1 will repeat several programmable times, followed by one sequence of bit map 2. The used length of map1 and map2 is programmable. In the event that map2 is programmed to a length of 0, then map2 is unused.

CPRI transmission rule doesn't use Modulo rule, which is used for OBSAI as a base rule of DBMR, instead CPRI uses fixed size packet data pattern and each link has its own DBM rule, so it doesn't need to use index or link number setup like OBSAI case. CPRI also uses channel LUT 0 ~ 5 for each link and only use 128 rules from each LUT instead of using 512 rules for OBSAI.

7.9.3 PE Pack

PE pack mechanism for OBSAI and CPRI is same to that in PD. MMR definition is also looks same to PD. To get more detailed information about this, please see section 7.8.2 in PD pack module part.

7.9.4 PE Timer

PE has an internal timer that paces with the clock at the natural PHY rate keeping pace with PhyT. In OBSAI's case, single circuit is multiplexed between links but CPRI has six separate circuits and single dual bit map rule per link.

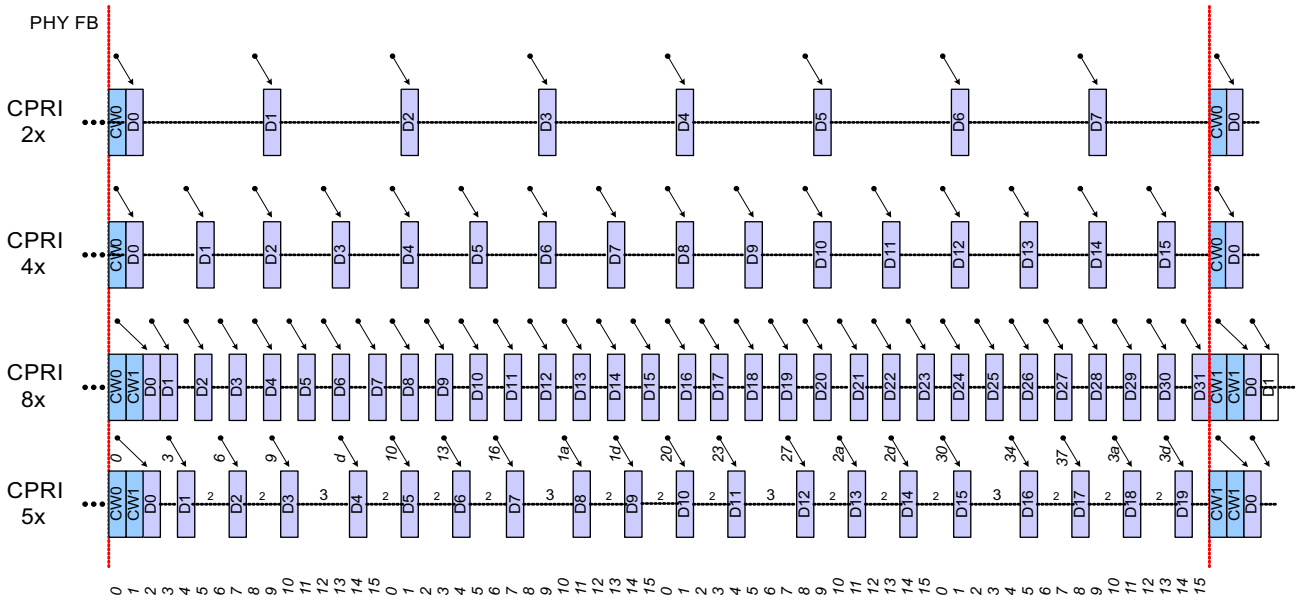
PE link starts up with AT phyT sync or RT sync and shut downed by stopping link on Phy frame boundary and RT sync.

[6]PE_LINK[1].TDD_AXC: AIF2 is tolerant of whole symbols of missing data on AxC-by-AxC basis. In TDD, SW is permitted to have gaps of whole symbols. Also support for Base Band hopping of GSM antenna carriers. SW manages which antenna symbol is sent to and which symbol is blank. If this value set to zero, the gap of TDD could be the whole symbols. For Egress DL setup, UL slots could be the gap of TDD DMA and user can only push the data into Tx queue only for DL slot time. If PE can not find any data from DB FIFO, it will automatically make the frame empty for UL slot time. It is user's responsibility to DMA DL data only for DL slot time.

[6]PE_LINK[1].DIO_CPPI: All AxC data within the link are treated as DIO traffic but Packets or control words are still remain as Multicore Navigator traffic.

[6]PE_LINK[1].DELAY: time delay between DB read and PE processing. Gives time for DB RAM fetch prior to message construction. It is normally programmed at 28 for OBSAI and zero for CPRI.

Figure 7-26. PE_TM_CPRI: Sample Rate Per Link Rate



7.9.5 PE Frame

A framing counter is maintained for each of 128 possible streams (when in AxC mode). The configuration of the frame counter is MMR configurable. The start of the timers is dependent on the timing of each AxC.

The Radio frame boundary is used as a timing reference point. PE will start AxC generation after AXC_OFFSET samples have passed. After start, counters will simply count without resynchronization.

PE_AXC_OFFSET [128]: Exact offset of an AxC. It is programmed in number of sys clock relative to the radio frame boundary (OBSAI) or phy frame boundary (CPRI).

PE_GLOBAL_EN_SET: Write-only function globally enabling PE.

PE_GLOBAL_EN_CLR: Write-only function globally clearing/disabling PE.

PE_GLOBAL_EN_STS: Read-only status of global enable state.

7.9.5.1 AxC Frame Counter

There are three main differences:

- PD/PE maintains timer count values independently for each of 128 channels
  - AT maintains three {RadT, RadT_UL, RadT_DL}
- PD/PE timers increment based on samples received
  - AT: counts clock cycles
  - PE:
    - OBSAI: count groups of 16 bytes (that is, four samples)
    - CPRI: count groups of four bytes (that is, one sample)
- PD/PE (LTE requirement) simultaneously supports six different sets of terminal counts.
  - AT: supports only one set of terminal counts

In support of the six different sets of LTE timing, most terminal count registers have six different versions. **PE_DMA0CHAN [128].FRM_TC**: programs on an AxC-by-AxC basis which of the six versions to use. The one exception of this register duplication is the **PE_FRM_MSG_TC [128].FRM_MSG_TC** which is so large that it is used to support all six versions in its 128 deep buffer. LTE only requires six or seven FRM_MSG_TC per version of the LTE counters; the user partitions the 128 deep buffers by dictating a start count (FRM_INDEX_SC) and a terminal count (FRM_INDEX_TC) for each of the six LTE possible counters.

The FRM_INDEX is a counter that parallels the FRM_SYM count, but programmed to wrap one or several times prior to the FRM_SYM wrap. LTE uses specifically six or seven counts (instead of the 120 or 140 count of the FRM_SYM. This FRM_INDEX concept is useful in limiting the supported range of **PE_FRM_MSG_TC[128]**.

7.9.5.2 Channel and Packet State

For each of 128 possible channels, a state bit is maintained indicating whether the channel is on/off or “in packet” or “out of packet”. This state information is used in global shutdown to indicate when it is safe to ack clock stop request. The state is also used to determine when a packet channel may transition into an OFF state.

Pkt Mode, In/Out packet is clear. When a SOP is received, the channel is IN_PKT. Once the EOP is received, the channel is OUT_PKT.

In AxC Modes (all radio standards other than WCDMA) the OFDM Symbol or GSM timeslot is considered packet boundaries and determines the IN/OUT_PKT state.

PE_CHAN_STS [4]: Status bit per channel indicating if the channel is ON or OFF.

PE_PKT_STS [4]: Status bit per channel indicating if the channel is IN/OUT_PKT. SOP or any Multicore Navigator beginning of packet sets bit. EOP or any closure of Multicore Navigator packet clears the bit. (For WCDMA, this bit is unused and set to 0x0).

7.9.6 PE DB Interface

7.9.6.1 CPRI Data Request I/F

CPRI consumes data on a word-by-word basis, where the DB passes data on a qword basis. The challenge in CPRI usage of AIF2 is supply enough data for each given channel without stealing DB bandwidth from competing channels. The solution involves a delicate balance of toggling between over and under subscribing requests.

The natural consumption rate is one qword every four CPRI data phases (per channel).

There are some exceptions to this rule:

- First two cycles of Ethernet (header append)
- 4B/5B packet encoding, 25% encoding overhead typical, 90% max
- Null Delimiter, one cycle hiccup every so often

At the beginning of time, when the channel is just starting up, the data requester checks every data phase (of a given channel) if data is available so it can start the transfers. The objective is to be slightly over subscribed, so that the buffer for that channel can fill to a comfortable level, but only slightly oversubscribed as to not rob too much bandwidth from adjacent channel activity. (25% over/under-subscribed is chosen basically because it’s easy to implement) Once the buffer has reached a comfortable full level (indicated by a programmable water mark), the circuit resorts to under-subscribing. In this way, the circuit oscillates between over/under subscription yielding a dither of actual available data in the buffer

PE_IN_FIFO [128].MF_WMARK: Message Construction FIFO Water mark. Water mark controls fill level for which lower/higher fetch rate is issued.

PE_IN_FIFO [128].MF_FULL_LEV: Message Construction FIFO full level, Set full level per channel. Full level controls halt of data fetch to prevent overflow.

Water marks should be set according to the parameters listed in Table 7-13. (For packet traffic, use the setting that closest matches the radio standard sampling rate):

Table 7-13. PE_DB_IF AxC Watermark Settings

Watermark	Over Full	Radio Standard
4 + 1	8 + 1	LTE 80MHz
2 + 1	4 + 1	LTE 20 MHz
1 + 1	1 + 1	WCDMA
1 + 1	2 + 1	Others

Table 7-14. PE_DB_IF Packet Watermark Settings

Watermark	Over Full	Radio Standard
1	2	2x or 4x CPRI CW Packet Channel
2	3	8x or 5x CPRI CW Packet Channel
Same to AxC table above	Same to AxC table above	Packet traffic in payload, use bandwidth analysis, Using AxC bandwidth value closes to Packet traffic

PE_IN_FIFO [128].SYNC_SYM: It marks the first position of non-zero TDD symbol of the frame. HW checks the first non-zero symbol of frame for startup sync with DMA. If 0th symbol is zero'ed and 1st symbol has data in TDD mode, then this value should be 1.

7.9.6.2 OBSAI Data Request I/F

Internally, PE is not oversubscribing OBSAI requests, but rather is requesting at the natural qword consumption rate of the interface. OBSAI mode does not present a peek bandwidth issue of DB requesting as CPRI could and therefore, OBSAI requests are handled with a much simpler approach. The same circuitry that is used for CPRI is also used for OBSAI with the exception that the filling and draining of the Data Request FIFO will be equivalent to the FIFO only having a depth of 1.

7.9.6.3 DIO Data Request I/F

The general operation is for DIO traffic to be packetized and posted into the DB FIFO. The PE prefetches qwords per channel in CPRI mode and fetches on-demand in OBSAI mode.

DIO traffic typically starts in advance of PE channel operation.When DIO traffic is running without PE channel operation running, DB continually flushes its FIFO content, every time the DIO addressing wraps. Additionally, DB give a control signal to PE in order for PE to flush it’s prebuffer as well. Once channel operation begins, the channel pull rate and DIO supply rate are equal yielding no more FIFO overflow concerns.

7.9.6.4 PE DB Token Request I/F

Tokens are a channel-by-channel trigger to DMA data into the DB buffer. Tokens are issued at twice the natural consumption rate of the channel, but later, tokens are filtered by the availability of DB buffer space to accept the DMA transfer. The result is a fast preload followed by a fairly constant DMA rate that mirrors the consumption rate. (Each token nominally triggers a DMA of four qwords for the given channel.)

There are differences between CPRI and OBSAI mode for token generation:

- OBSAI: PE Frame maker visits each channel once per OBSAI message (or once per qword of consumption).
- CPRI: PE Frame maker visits each channel once per sample (or four times per qword of consumption).

Simply put, the token generator is a divider, issuing a token every other OBSAI channel ready signal or every 8th CPRI channel ready signal. But there is a question of phase alignment. The phase alignment is programmable which gives some control over system minimum DMA latencies.

PE_GLOBAL [1].TOKEN_PHASE: Adjusts phase of token issue. OBSAI 50% phase adjustment. CPRI 12.5 percent phase adjustment. It is used to adjust minimum DMA transfer latencies. OBSAI: only lsb is used. Zero means no latency and one means 50 percent phase latency. CPRI: all three bits give 8th phase alignment. Zero value corresponds to start of issue on radio frame boundary. this phase adjustment only used when DMA time condition from PKTDMA is very tight.

In a startup condition, the channel is not ON when tokens are first issued. For this startup condition, random phase of token issue is performed. Once the channel starts up, PE_FM gives indication that the channel is at the radio frame boundary. This resets the token issue phase. A token phase of 0 will begin issue at the radio frame boundary, then every 2x or 8x (depending on CPRI vs. OBSAI). Every subsequent radio frame boundary will again resync the token issue.

7.9.7 PE Message Construction

7.9.7.1 PE_MC_ENET

No 32-bit data is required in the first two data phases of a packet. The ENET circuit maintains state bits that both control the insertion of the ENET header and preamble and suppression of data pull operations for these data phases.

- Preamble: Constant value: 8'b1010_1010
- Start of Frame: Constant value: 8'b1010_1011

CRC is not calculated over Ethernet Preamble and SOF. PE_MC_ENET provides control signals to PE_MC_CRC which prevent CRC from being calculated over these two words

PE_GLOBAL [1].ENET_HDR_SEL: Adjusts bit order of Ethernet preamble and start of Frame. 0: 0xAAAAAAAB, 1: 0x555555D5

PE_DMA0CHAN [128].ETHERNET: If this bit is set to one, this channel is Ethernet and this field controls insertion of the Ethernet preamble and SOF.

7.9.7.2 PE_MC_CRC

The PE_MC_CRC performs CRC calculation for up to 128 channels. It serially processes up to four bytes of a single channel every clock cycles. From the perspective of the CRC circuit, the channels are serviced in random order so that the circuit maintains state context for all 128 channels. The input is a four-byte interface which is an SOP and EOP indicator. All four-byte lanes are fully populated for all but the last EOP input which can be partially populated depending on CRC format to be calculated.

The user is constrained to have placed a gap in the input data. The pull engine will pull the last word containing the CRC gap; PE_MC_CRC will calculate the CRC and insert it into this gap. The user is further constrained to choose a packet length such that the CRC gap is wholly contained within a single 32-bit word. Assuming these constraints, the CRC circuit does not need predictive packet length and can simply insert the CRC in the EOP data phase of the packet.

The PC_ME_ENET circuit will control PC_MC_CRC to suppress CRC calculation over Ethernet preamble and SOF.

7.9.7.2.1 CPRI

[6]PE_CRC.CRC_8POLY: Programmable polynomial for CRC8

[6]PE_CRC.CRC_8SEED: Programmable seed value for CRC8

7.9.7.2.2 OBSAI

PE_DMA0CHAN [128].CRC_EN: CRC enable, disable. CRC will be checked on AxC by AxC basis.

PE_DMA0CHAN [128].CRC_TYPE: CRC Type (8-bit, 16-bit, 32-bit)

PE_DMA0CHAN [128].CRC_HDR: CRC 16 is calculated over OBSAI header. This field only valid for OBSAI type "CONTROL" and "MEASUREMENT"

7.9.7.3 PE_MC_OBSAI_HEADER

PE_OBSAI_HDR [128].OBSAI_TS_ADR: OBSAI header (5:0) TS: only used if TS is inserted instead of generated

PE_OBSAI_HDR [128].OBSAI_TYPE: OBSAI header (10:6) TYPE: inserted into OBSAI header TYPE field

PE_OBSAI_HDR [128].OBSAI_ADR: OBSAI header (23:11) ADR: inserted into OBSAI header ADDRESS field

PE_OBSAI_HDR [128].OBSAI_TS_MASK: This indicates how many bits of the TS should be compared or calculated

PE_OBSAI_HDR [128].OBSAI_TS_FRMT: OBSAI timestamp generation algorithm

PE_OBSAI_HDR [128].OBSAI_PKT: Select if this channel is used for OBSAI AxC data or Packet data

PE_OBSAI_HDR [128].BB_HOP: If this bit is on, the OBSAI Address is taken from Multicore Navigator protocol specific data instead of obsai_adr mmr bits. This option is used for GSM baseband hopping.

7.9.8 PE RT Interface

The RT interface is basically a 128deep x (data word + control signals) FIFO. The output of the FIFO is basically the QBI (Quad Byte Interface) format which feeds the RT. There are six copies of this FIFO, one per link. RT controls pulling from the FIFO.

The depth of the FIFO has two basic functions:

- Allows the user some amount of timing slop in configuring PE timing
- Allows for timing slop between ideal and actual link timing
  - OBSAI control messages, headers, K-Char and channel interleaving

7.9.8.1 PE_RT_SYNC

Under clock stop conditions, the PE stops giving contribution and RT continues operations without PE contribution. This transition from PE_on to PE_off only occurs on a PHY frame boundary. The use of this signal by RT is simple: It only looks at this signal when the data represents a Frame Boundary

- If CI and PE streams are aggregated (WCDMA) or merged (OFDM), CI data dictates Phy Frame Boundary
- If there is only PE stream, the PE FB signals dictates

So, turning off the signal is fairly trivial. Once the shutdown protocol has turned off all channels, the PE_RT_SYNC is deasserted.

7.10 DB (Data Buffer)

The DB (Data Buffer) is the main data path buffer for AIF2. It consists of two independent subsystems, the Ingress DB (IDB) and the Egress DB (EDB). Data in the IDB is transferred to other data resources via the VBUSM interface. Data from other resources is transferred into the EDB via the VBUSM interface.

DB supports the following features:

- 16 byte (Quad word) interface
- Supports mix of FIFO (packet data) and Circular Buffers (DirectIO)
- Supports up to 128 buffer channels (AxC's and packet mode flows)
- FIFO size is programmable per-buffer channel (with limitations)
- Circular Buffer for DIO sizes selectable between 128 and 256 bytes
- Data Trace Support. Formats data trace data and framing data from RM into 128-bit quad words
- Per-buffer channel programmable data swapping
- Per-buffer channel programmable IQ ordering
- Data RAM – 1K x 16 byte

- Sideband data RAM – 1K x 24 bit
- Data Trace RAM - 160 x 16 byte
- Debug Support Provided generation debug data (QW) and monitoring data offset.

The IDB and EDB can be segmented into a maximum of 128 Buffer channels. These channels can be a mix of AxC packet data, non-AxC packet data, and DirectIO data. For packet data, the DB is segmented into FIFOs of programmable size with the restrictions indicated below (basic trade off between number of FIFOs and depth per FIFO). For DirectIO, the DB is segmented into either 128 or 256 byte circular buffers via **DB_IDB_CFG.DIO_LEN. And DB_EDB_CFG.DIO_LEN**.

7.10.1 Addressing

Although the size of each FIFO in the DB is programmable, only a limited number of sizes will be supported. The minimum size is eight quad-words or 128-bytes. All of the other sizes are power of two multiples of eight quad-words up to a maximum of 256 quad-words or 4K bytes. This scheme is detailed in Table 7-15.

Table 7-15. DB Supported FIFO Buffer Sizes

Supported FIFO Buffer Sizes (Quad-Words)	Maximum Number of Buffer Channels Supported	BUF_DEPTH[2:0]
8	128	0
16	64	1
32	32	2
64	16	3
128	8	4
256	4	5,6,7

The data out of the DB is transferred in bursts of up to four quad-words. For FIFO requests, the AD provides the same channel id for each quad-word of the burst. For Direct IO requests, each quad-word of data can be sourced from a different buffer channel number. The AD orchestrates this by providing the channel id and address for each quad-word of the burst.

The size and starting location of each individual FIFO and the starting location of each individual circular buffer are programmable via per-buffer channel pointer registers as shown in Table 7-16. For FIFO buffer channels, The DB contains logic that uses the data in these registers to calculate the read and write pointers for each FIFO buffer channel. This requires that the DB also store an internal offset within the FIFO buffer channel where the next entry will be written to or read from. The offset value increments each time an entry is accessed and will wrap back to 0 when the end of the buffer is reached.

The IDB and EDB contains enough storage for up to 128 offsets in separate internal registers for read and write accesses. An offset is automatically cleared to 0 when a disabled buffer channel is enabled via its associated buffer channel enable bit.

The same pointer registers are used for both reading and writing the DB RAMs. However, because the DB RAM is two-ported, each buffer channel has separate internal read and write offset registers.

For circular buffers, the buffer depth is provided by DB_IDB(EDB)_CFG.DIO_LEN, so the DB_IDB(EDB)_PTR_CHxx.BUF_DEPTH field is ignored. Also, because the address of the data within a circular buffer is provided by the PD for writes and the AD for reads, the internal offset registers are not used, as well. This allows random access within the circular buffer on a quad-word basis.

Table 7-16. DB_IDB(EDB)_PTR_CH Parameters for Programming Buffer Location and Size

Parameter	Width	R/W	Description
BASE_ADDR	7	R/W	7 MSBs of starting address of buffer channel FIFO or Direct IO buffer. Lower 3 bits are forced to 0 internally (working based on 8 Quad word)
BUF_DEPTH	3	R/W	Encoded buffer depth (8, 16, 32, 64, 128, 256)

The **DB_IDB(EDB)_PTR_CHxx** registers are always accessible to the system via the VBUSP configuration bus. Reading of these registers will not affect the internal operation of the DB.

Example:

To configure buffer channel 2 of the IDB as a 128-byte FIFO located at address 16 (QW), the DB_IDB_PTR_CH2 register is programmed as shown below:

- BASE_ADDR = 0x02
- BUF_DEPTH = 0x0 (8 QW)

The data out of the DB is transferred in bursts of up to four quad-words. For FIFO requests, the AD provides the same channel id for each quad-word of the burst. For Direct IO requests, each quad-word of data can be sourced from a different buffer channel number. The AD orchestrates this by providing the channel id and address for each quad-word of the burst.

7.10.2 Alignment Logic

The DB also has logic that provides a Data Swapper that performs programmable endian independent data and IQ swapping logic on a per-buffer channel basis. This logic is controlled by the **DB_IDB(EDB)_CFG_CHxx.DAT_SWAP** and **DB_IDB(EDB)_CFG_CHxx.IQ_ORDER**. These fields are described in Table 7-17.

Table 7-17. DB_IDB_CFG_CHxx Parameters for Programming Alignment

Parameter	Width	R/W	Description
DAT_SWAP	2	R/W	Data swapping schemes: 00 – no swap 01 – byte swap 10 – ½ word swap (16-bit swap) 11 – word swap (32-bits)
IQ_ORDER	2	R/W	IQ swapping scheme: 0x – no swap 10 – byte swap 11 – 16-bit swap

The internal format of the data before the data swapper is big endian because the PD is natively big endian and this is the same to PE and egress data swapper.

AIF2 IQ position and endian mode change (by data swapping) is 100% programmable by modifying above fields. In case of AIF1, IQ position is not as flexible and there is no data swap option. The problem that user may experience is core pac specifies a complex multiply instruction that basically says that IQ should not be swapped between BE & LE and some other module's definition could be different and it might cause non-consistencies about handling of IQ swap for endianess. As a result, highly flexible AIF2 configuration was made for users to handle any reasonable endian definition.

PD and PE is shuffling the data to make it fit into its data transmission policy. User may see the slight difference about this shuffling between AIF1 and AIF2. In case of AIF1, PD, PE sends or receives odd sample first and even sample last but AIF2 do the opposite way. (even sample first, odd sample last) User can not get same data pattern by manipulating IQ swap or data swap field, but application program can change the sample order of RAC process to avoid this kind of confusion.

7.10.3 Buffer Channel Enables

Each of the up to 128 buffer channels has an individual enable bit in the IDB and EDB. These enables are split among four registers, as shown in Table 7-18. These enables are used by the IDB to drop any data that the PD writes to a disabled buffer channel, and also clear the read and write pointer offsets, as discussed above. The PD data is dropped as it exits the PD-to-DB Bridge at the VBUS clock rate.

Table 7-18. Mapping of IDB Buffer Channel Enables

Register	Buffer channel Enables
DB_IDB(EDB)_CH_EN0	31:0
DB_IDB(EDB)_CH_EN1	63:32
DB_IDB(EDB)_CH_EN2	95:64
DB_IDB(EDB)_CH_EN3	127:96

7.10.4 Other Functionality

7.10.4.1 Protocol-Specific Data

PKTDMA supports the addition of protocol specific data to a packet to non-payload portions of the Multicore Navigator packet. For AxC packet data, AIF2 can be programmed to add the AxC number, symbol number, an ingress/egress identifier, and a AIF2 protocol identifier by setting **DB_IDB_CFG_CHxx.PS_EN** to a 1. This field is only meaningful for channels that store AxC packets and is ignored for Direct IO buffers. Buffer channels that store non-AxC packet data must have this bit set to 0.

The EDB can identify protocol specific data via the *data_type* sideband signal provided by the PKTDMA. When it detects protocol specific data, the EDB stores it as sideband data in the Egress DB RAM.

7.10.4.2 Packet Type

Multicore Navigator also supports a five-bit packet type field that is inserted into the packer descriptor. The packet type for each buffer channel, which is sent to the AD as sideband data, is programmed via **DB_IDB_CFG_CHxx.PKT_TYPE**.

7.10.4.3 Egress DIO offset

DIO_OFFSET field added to all 128 of the Egress Data Buffer Channel Configuration Registers. It is a per-channel offset used for the DIO packetization process. The value in this field is the DIO buffer address (QW level) from the DIO DMA Engine that is the SOP of the DIO packet. This value is closely related to the AxC offset value. In case of OBSAI, this offset should be the same to the External AxC offset value for that channel.

7.10.4.4 EDB Packet Mode Control

This field takes care of the CPRI packet starvation from AD_ESCH_CFG.pri selection (see AD section for more info). When set to a 1, the non-AxC Multicore Navigator packet tokens (PM tokens) are sent to the same Token FIFO in the DB as the AxC Multicore Navigator packet tokens. Otherwise, the tokens are put into separate FIFOs

7.10.4.5 Consumption-Based Scheduling

For packet data, the PE controls the scheduling of the data transfers to the EDB via the data transfer tokens it issues. These tokens are issued on a consumption basis. Each time the PE consumes 32 bytes of data from the EDB for a given buffer channel or detects an EOP for that channel, it issues a data transfer token to the EDB regardless of the occupancy of the data buffer channel. Because each token can move up to 64 bytes of data (that is, only exceptions are EOP, SOP, and FLUSH tokens), the PE issues tokens at twice the consumption rate. The FIFO Occupancy Manager keeps track of the number of entries in each FIFO, and it will discard a token for the following reasons:

- FIFO has less than 64-bytes of available space
- Buffer channel is not enabled
- AxC token and AxC Token FIFO full (that is, also includes flush tokens)
- Packet mode token and Packet Mode Token FIFO full

7.10.5 DB Debug

7.10.5.1 Overview

From a debug perspective, it may be desirable for the user to verify that data has been DMA'ed correctly to/from the AIF2 DB. As such, the AIF2 provides special debug resources to allow a user to do the following using MMRs that are accessible via the VBUS Config interface:

Write data to the Ingress DB RAM for a particular DMA channel and have that data transferred from the AIF2 via Multicore Navigator or Direct IO DMA to the desired destination (that is, L2 memory).

Read data transferred to the Egress DB RAM via Multicore Navigator or Direct IO DMA for a particular DMA channel from a source location (that is, L2 memory).

The AIF2 must be in a static (idle) state when these debug transfers are performed. Both the PE and PD must be disabled, but the AIF2 could be configured in "bypass mode" to keep the daisy chain active.

The debug schemes are internally implemented in such a way as to make the most use of the existing data transfer protocols when accessing the DB memory via the VBUS Config interface. For writing the Ingress DB memory, the debug logic takes over the PD interface to the Ingress DB. For reading the Egress DB memory, the debug logic takes over the PE interface to the Egress DB. The Multicore Navigator and Direct IO DMA operate exactly the same for debug mode and have no knowledge that the AIF2 is configured for debug mode operation. As such, the PKTDMA Controller should be configured in a normal manner when using debug mode.

7.10.5.2 Ingress DB Debug

DB_IDB_CFG.IDB_DEBUG_EN is set to "1" to put the AIF2 into Ingress debug mode. In this mode, all PD channels must be disabled to prevent data from being written to the IDB.

The AIF2 provides MMRs that are accessible via the VBUS Config interface to implement the Ingress DB debug scheme as described below. The debug scheme is to indirectly write the payload data and sideband data used for control into the Ingress DB RAM. The payload data and sideband data are first written into holding MMRs and then another MMR is written to that initiates the transfer of this data into the RAM. This procedure is used to write to buffers configured as FIFOs for packet data or circular buffers for DirectIO data These MMRs are described below:

Because the Ingress DB memory is 128-bits wide, but the VBUS Config interface is only 32-bits, four write data MMRs are provided (that is, **DB_IDB_DEBUG_D3**, **DB_IDB_DEBUG_D2**, **DB_IDB_DEBUG_D1**, and **DB_IDB_DEBUG_D0**)

The **DB_IDB_DEBUG_SBND** MMR provides the signals that are used to emulate the PD/DB interface (e.g. SOP, EOP). Of particular note are **DB_IDB_DEBUG_SBND.DIO_WR_EN** and **DB_IDB_DEBUG_SBND.FIFO_WR_EN** which distinguish whether the buffer being written to is configured as a FIFO or a circular buffer. For FIFO buffers, the address where the data is written in the RAM is provided via **DB_IDB_DEBUG_SBND.CH_ID** and an internally generated offset value that requires no user control. For circular buffers, the address is provided via **DB_IDB_DEBUG_SBND.CH_ID** and **DB_IDB_DEBUG_SBND.DIO_ADDR**.

Once all of the five MMRs mentioned above have been loaded, the data in those MMRs is written to the Ingress DB RAM when **DB_IDB_DEBUG_DB_WR** is set to 1. Note that each entry in the DB RAM consists of both payload data and sideband data.

The method of transferring the debug data written to the Ingress DB RAM out of the AIF2 varies depending on whether it is packet data or DirectIO data. For packet data, the IDB has to generate a packet data token to transfer the data to L2, as an example, via Multicore Navigator. Tokens are automatically generated in the IDB by either writing 64 bytes of data to a data buffer channel or creating an end of packet condition by setting **DB_IDB_DEBUG_SBND.EOP** to 1.

For DirectIO data, the data is moved to L2 via a timing strobe created by the AT. In this case, all of the data that will be transferred to L2 must be indirectly written to the Ingress DB RAM first. The timing strobe is then created by writing a 1 to the **AT_INTERNAL_EVT_FORCE.EVT_FORCE[11:0]** bit that creates the desired event to the AD Prior to creating the timing strobe via the MMR write, the automatic generation of the strobe must be turned off by writing a 0 to the same bit number in **AT_INTERNAL_EVT_ENABLE.EVT_EN[11:0]** Although creating the timing strobe kicks off the data transfer process, the data buffer channels that are transferred to L2, as an example, are selected by programming MMRs in the AD DIO Controller

7.10.5.2.1 Packet Data Example

Table 7-19 shows an example of how an 88-byte packet is written into data buffer channel 5 of the Ingress DB RAM for debug. The table shows how key fields of **DB_IDB_DEBUG_SBND** are written in order to create packet data transfer tokens. All other fields of these MMRs are assumed to be 0 for this example. Each write of the two sideband MMRs must also be accompanied by writes to **DB_IDB_DEBUG_D[3:0]**. For this example, **DB_IDB_DEBUG_SBND.DIO_WR_EN = 0** and **DB_IDB_DEBUG_SBND.FIFO_WR_EN =1**.

Table 7-19. Debug Sideband Data Example for Packet Data

SOP	EOP	CH_ID	XCNT	Comments
1	0	0x05	0x10	Start of Packet
0	0	0x05	0x10	
0	0	0x05	0x10	
0	0	0x05	0x10	Token created
0	0	0x05	0x10	
0	1	0x05	0x8	End of packet; token created; only 8 bytes valid

7.10.5.2.2 DirectIO Example

Table 7-20 shows an example of how 64 bytes of data is written into data buffer channel 16 of the Ingress DB RAM for debug. The table shows how key fields of **DB_IDB_DEBUG_SBND** are written. All other fields of these MMRs are assumed to be 0 for this example. Each write of the two sideband MMRs must also be accompanied by writes to **DB_IDB_DEBUG_D[3:0]**. For this example, **DB_IDB_DEBUG_SBND.DIO_WR_EN** = 1 and **DB_IDB_DEBUG_SBND.FIFO_WR_EN** =0.

Table 7-20. Debug Sideband Data Example for DirectIO Data

CH_ID	XCNT	DIO_ADDR
0x11	0x10	0x0
0x11	0x10	0x1
0x11	0x10	0x2
0x11	0x10	0x3

In this case, **DB_IDB_DEBUG_SBND.CH_ID** provides the buffer in the Ingress DB RAM and **DB_IDB_DEBUG_SBND.DIO_ADDR** provides the location within the buffer where the data is written.

7.10.5.3 Ingress DB Activity Monitoring

The normal activity of the IDB can be monitored by reading the per-channel internal offset values used in creating the read and write pointers to the buffer channels. If the offset value for a particular channel is changing, then there is activity on that channel. There are separate read and write pointer offset values for each channel, and each offset is accessed independently. The method of reading these offset values is described below:

The seven-bit “write offset” address (e.g. the channel 0x5 offset is at address 0x5) is written to **DB_IDB_DEBUG_OFS.WADDR**. The “write offset” can be read from **DB_IDB_DEBUG_OFS_DAT.WOFF**.

The 7-bit “read offset” address is written to **DB_IDB_DEBUG_OFS.RADDR**. The “read offset” can be read from **DB_IDB_DEBUG_OFS_DAT.ROFF**.

7.10.5.4 Egress DB Debug

DB_EDB_CFG.EDB_DEBUG_EN is set to “1” to put the AIF2 into Egress debug mode. In this mode, all PE channels must be disabled to prevent all tokens and read requests from the PE.

7.10.5.4.1 Packet Data Debug

For packet data, the debug scheme involves transferring a packet in L2 memory, as an example, to the EDB via PKTDMA and then indirectly reading it via the VBUS Config interface. The CPU writes the packet into a buffer in L2 memory and adds a Multicore Navigator packet descriptor. The EDB then creates packet data tokens via an MMR that is accessible via the VBUS Config interface that schedules 64-byte data transfer requests for that packet. Finally, the payload data and sideband control data for the packet are read to confirm that the proper data was transferred to the EDB.

The MMRs are described below:

- Because the Egress DB memory is 128-bits wide, but the VBUS Config interface is only 32-bits, four read data MMRs are provided (that is, **DB_EDB_DEBUG_D3**, **DB_EDB_DEBUG_D2**, **DB_EDB_DEBUG_D1**, and **DB_EDB_DEBUG_D0**).
- The **DB_EDB_DEBUG_SBND** MMR provides the key sideband control data that accompanies each data entry in the Egress DB RAM (e.g. SOP, EOP) and is used as control to the PE.
- The **DB_EDB_DEBUG_DATA_TOK** MMR is used to provide 64-byte packet data tokens that the Multicore Navigator controller uses to move data from L2 to the AIF2.
- The **DB_EDB_DEBUG_RD_CNTL** MMR provides the location of the data to be read.
- The **DB_EDB_DEBUG_DB_RD** MMR is used to initiate the read of the Egress DB RAM and also transfer the tokens to the Token FIFO in the EDB.

Example:

Transferring an 80-byte packet that already exists in L2 memory to buffer channel 24 of the Egress DB RAM and reading it is shown below:

1. Create a packet data transfer token for Packet PKTDMA channel 24 by writing **DB_EDB_DEBUG_DATA_TOK.TOKEN_CH** = 0x18.
2. Issue token by writing **DB_EDB_DEBUG_WR_TOK**. This token will move the first 64 bytes of packet data to buffer channel 24 of the Egress DB RAM via PKTDMA.
3. Repeat steps above to issue the second token. Because the packet is only 80 bytes, this token only returns 16 bytes of data.
4. Wait for all of the Multicore Navigator transfers to complete (say 10µsec)
5. Setup the location to read data from in the Egress DB RAM by writing **DB_EDB_DEBUG_RD_CNTL.FIFO_RD_EN** = 0x1 and **DB_EDB_DEBUG_RD_CNTL.CH_ID** = 0x18. All other values of **DB_EDB_DEBUG_RD_CNTL** have to be 0.
6. Initiate the read of the first 16-bytes of packet data and associated sideband data from the Egress DB RAM and subsequent loading into the holding registers by writing **DB_EDB_DEBUG_DB_RD**.
7. Read first 16-bytes of data and accompanying sideband control data via **DB_EDB_DEBUG_D[3:0]** and **DB_EDB_DEBUG_SBND**.

8. Repeat steps 6 and 7 four more times to read the remaining 64-bytes of data and accompanying sideband control data.

7.10.5.4.2 Direct IO Debug

For DirectIO data, the debug scheme involves transferring data in L2 memory, as an example, to the EDB via the DirectIO hooks in the PKTDMA Controller and then indirectly reading it via the VBUS Config interface. The CPU writes all of the data that will be transferred to the AIF2 into buffers in L2 memory. The data is transferred to the AIF2 via a timing strobe created by the AT. The timing strobe is then created by writing a 1 to the **AT_INTERNAL_EVT_FORCE.EVT_FORCE[11:0]** bit that creates the desired event to the AD. Prior to creating the timing strobe via the MMR write, the automatic generation of the strobe must be turned off by writing a 0 to the same bit number in **AT_INTERNAL_EVT_ENABLE.EVT_EN[11:0].** Although creating the timing strobe kicks off the data transfer process, the data buffer channels that the L2 data, as an example, are transferred to are selected by programming MMRs in the AD DIO Controller.

Finally, the payload data and sideband control data for the packet are read to confirm that the proper data was transferred to the EDB. The same MMRs described above for debug of packet data are used fro DIrectIO data.

Example:

Transferring 64-bytes of DirectIO data from L2 memory to buffer channel 48 of the Egress DB RAM and reading it is shown below. Prior to step one the AD DIO controller must be programmed to transfer the L2 data to the desired data buffer channels in the EDB.

1. CPU creates AT timing strobe as described above.
2. Wait for all of the Multicore Navigator transfers to complete (say 10usec)
3. Setup the location to read data from in the Egress DB RAM by writing **DB_EDB_DEBUG_RD_CNTL.DIO_RD_EN** = 0x1 and **DB_EDB_DEBUG_RD_CNTL.CH_ID** = 0x30. All other values of **DB_EDB_DEBUG_RD_CNTL** have to be 0.
4. Initiate the read of the first 16-bytes of DirectIO data and associated sideband data from the Egress DB RAM and subsequent loading into the holding registers by writing **DB_EDB_DEBUG_DB_RD**.
5. Read first 16 bytes of data and accompanying sideband control data via **DB_EDB_DEBUG_D[3:0]** and **DB_EDB_DEBUG_SBND**.
6. Repeat steps 3 through 5 three more times to read the remaining 48-bytes of data and accompanying sideband control data.

NOTE: Because writing this data into the Ingress DB RAM uses the actual control logic, it is recommended that the AIF2 be reset (that is, either soft or hard reset) before setting **DB_EDB_CFG.EDB_DEBUG_EN** to a 1 and after clearing **DB_EDB_CFG.EDB_DEBUG_EN** to a 0.

7.10.5.5 Egress DB Activity Monitoring

The normal activity of the EDB can be monitored by reading the per-channel internal offset values used in creating the read and write pointers to the buffer channels. If the offset value for a particular channel is changing, then there is activity on that channel. There are separate read and write pointer offset values for each channel, and each offset is accessed independently.

The method of reading these offset values is described below:

- The seven-bit “write offset” address (e.g. the channel 0x5 offset is at address 0x5) is written to **DB_EDB_DEBUG_OFS.WADDR**. The “write offset” can be read from **DB_EDB_DEBUG_OFS_DAT.WOFF**.
- The seven-bit “read offset” address is written to **DB_EDB_DEBUG_OFS.RADDR**. The “read offset” can be read from **DB_EDB_DEBUG_OFS_DAT.ROFF**.

The EDB also provides the additional activity monitors to facilitate debug using 24-bit EOP counter (**DB_EDB_EOP_CNT**) that increments each time EDB receives an EOP from the PKTDMA. This counter will wrap when it reaches its maximum value.

7.10.6 DB Data Trace Feature

The IDB has a separate 160x128 two-port RAM to support the Data Trace feature. This data is also sourced from the RM and destined to be transferred out of the AIF2 to L2 memory via the Multicore Navigator controller in DirectIO mode. The IDB controls the writing of the Data Trace Buffer and the AD controls the reading. A bridge is provided to interface the two clock domains mentioned above. The output path of this data is the same as that from the Data Buffer RAM.

The trace data comes from the RM as 16-bits (that is, even and odd bytes). The bytes are packed into 128-bit words in big endian byte format by the Data Trace Format/Control block shown in Figure 7-27 before being written into the Data Trace RAM.

For each byte of trace data, the RM provides three framing signals, K-Char indicator, 8B/10B code violation, and frame boundary. Only the K-Char indicator and 8B/10B code violation bits are stored in the Data Trace RAM. These signals are packed into 128-bit words in big endian format before being written into the Data Trace RAM. The 128-bit word contains the framing signals for 64-bytes of trace data as shown.

Figure 7-27. Data Trace Quad-Word Byte Alignment (D0 First in Time)

127	119	111	103	95	87	79	71	63	55	47	39	31	23	15	7
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15

Figure 7-28. Framing Data Quad-Word Alignment (S0 First in Time)

127	125	123		3	1
S0	S1	S2	. . . . .	S62	S63

The trace data and framing data information are treated as separate buffer channels and as such, are written into separate regions in the Data Trace RAM. The trace data is assigned the lower 128 locations in the Data Trace RAM and the framing data is assigned the upper 32 locations. As such, there are no programmable registers associated with the depth of the Data Trace RAM channels.

The Data Trace Format/Control block has several modes that are controlled by bits in the **DB_IDB_CFG** register as listed in Table 7-21.

Table 7-21. DB_IDB_CFG Parameters for Controlling Data Trace

Parameter	Width	R/W	Description
DT_EN	1	R/W	Data Trace Enable: 0- Clear data trace control logic in IDB and AD. This has the effect of flushing the contents of the Data Trace RAM and the Data Trace Token FIFO 1- Enable capture of trace data and/or framing data per settings of DTB_EN and DTF_EN
DTB_EN	1	R/W	Trace Data Capture Enable: 0- do not capture trace data 1- capture trace data
DTF_EN	1	R/W	Framing Data Capture Enable: 0- do not capture trace framing data 1- capture trace framing data
DT_SYNC	1	R/W	Data Trace Sync Enable: 0- capture data trace data without any synchronization 1- synchronize start of data capture to frame boundary signal from RM

When data trace capture is synchronized to the frame boundary, there are two possible alignment modes depending on whether the frame boundary occurred on odd or even bytes. The least significant byte of the first quad-word stored in the Data Trace RAM is always the K-char regardless of whether the first byte was the even or odd byte from the RM.

The **DB_IDB_CFG.DT_EN** bit acts as a soft reset for all of the data trace control logic in the IDB and AD when it is a “0”. No trace data or framing data will be captured while this bit is “0”. If it is ever set to “0” while data is being captured, the capture process will be immediately stopped and all data in the Data Trace RAM and the tokens in the Data Trace Token FIFO will be flushed. The data capture process will be restarted again when this bit is set to “1” according to the state of **DB_IDB_CFG.DTB_EN**, **DB_IDB_CFG.DTF_EN**, and **DB_IDB_CFG.DT_SYNC**.

If **DB_IDB_CFG.DTB_EN** and **DB_IDB_CFG.DTF_EN** are both disabled, then no data will be stored in the Data Trace RAM. If Data Trace is disabled after being enabled, the capture will terminate in the same way as it was programmed to start:

Synchronized: Data Trace Format/Control block will stop capture on next FB

Not synchronized: Data Trace Format/Control block will stop capture immediately.

The Data Trace Format/Control block posts data transfer tokens into the Data Trace Token FIFO. The token encoding depends on the values programmed into **DB_IDB_CFG.DTB_EN** and **DB_IDB_CFG.DTF_EN**. The tokens posted to the FIFO are two bits wide and the encoding is { **DB_IDB_CFG.DTB_EN**, **DB_IDB_CFG.DTF_EN**}

The output of the Data Trace Token FIFO goes to the AD Direct IO Controller which pops new tokens off of the FIFO. Each token result in a transfer of data from the AIF2 Data Transfer RAM to L2 memory. The amount of data depends on the token type described in Table 7-22. Turning data trace capture “off” causes the Data Trace Token FIFO to eventually become empty, resulting in no further DMA activity.

Table 7-22. Data Trace Token Types

DTF_EN	DTB_EN	Per-token Data Transfer Characteristics
0	0	Data trace feature is disabled
0	1	Only capturing trace data: Token posted after 64 bytes of trace data has been written into the Data Trace RAM
1	0	Only capturing framing data: Token posted after 16 bytes of framing data has been written into the Data Trace RAM
1	1	Capturing trace and framing data: Token posted after 64 bytes of trace data and 16 bytes of framing data have been written into the Data Trace RAM

7.11 AD (AIF2 DMA)

7.11.1 Ingress Scheduler (ISCH)

The ISCH receives requests to transfer data out of the AIF2 from the Ingress Data Buffer (IDB) and the internal AD DirectIO controller. The arbitration priority used by the Ingress Arbiter subblock is configured via **AD_ISCH_CFG.PRI**. When this bit equals “0”, the order is:

- DirectIO (highest priority)
- Packet data
- Data Trace (lowest priority)

When this bit equals “1”, the order is:

- Packet data (highest priority)
- DirectIO
- Data Trace (lowest priority)

The ISCH provides controls to the IDB for each quad-word of data that is moved out of the IDB. For packet data requests, the buffer channel number is the same for each quad-word of a burst. For DirectIO requests, both the buffer channel number and the address within each buffer channel can be different for each quad-word. For Data Trace requests, the buffer channel number may change within a burst if both trace data and framing data are moved as part of the burst.

The ISCH has logic to insert zeros to support up to two missing timestamps. It also uses the SOP_MISS and EOP_MISS sideband signals to determine if it also needs to assert the SOP or EOP signals to the PKTDMA.

Because the PD can fail packets for certain conditions (that is, CRC checksum failure and more than two missing timestamps, etc.), the ISCH can be programmed to either drop these packets via the PKTDMA or mark these packets as having an error via the PKTDMA. The response of the ISCH to a “failed” packet is determined by **AD_ISCH_CFG.FAIL_MODE** (0: drop; 1: mark).

When fail_mode is set for *mark*, the pkt_error[0] bit in the packet descriptor is set to 1 when a *failed* packet is received from the PD. The failed packet is handled like a good packet by the PKTDMA sub-system. The user application can detect that the packet was failed by looking at the pkt_error bits in the descriptor.

A 24-bit EOP counter (**AD_ISCH_EOP_CNT**) that increments each time the ISCH sends an EOP to the PKTDMA is provided as an activity monitor. This counter will wrap when it reaches its maximum value.

7.11.2 Egress Scheduler (ESCH)

The ESCH receives requests to transfer data to the AIF2 from L2 memory or another peripheral (e.g. TAC). The AIF2 is either configured such that the AxC data is either all packet data or all DirectIO data. There is no mixing of AxC data. Regardless of how the AxC data is handled, there can be non-AxC packet data flowing as well. The priority of this data is configured via AD_ESCH_CFG.PRI and it is totally different to Ingress scheduler.

- 0 means AxC requests (DIO and Multicore Navigator based) have higher priority than non-AxC packet requests and this should be used for OBSAI.
- 1 means Non-AxC packet requests have higher priority than AxC requests (DIO and Multicore Navigator based) requests and this should be used for CPRI.

When this bit is set to 1, non-AxC Multicore Navigator packets have higher priority than AxC packets. We had to add this capability for CPRI because once they are started, CPRI packets cannot be starved. We recommend that this bit be set to 1 for CPRI applications that have non-AxC packets. For OBSAI, we expect this bit to be 0, because OBSAI has a mechanism for handling non-AxC packet starvation.

DB register has DB_EDB_CFG.pm_ctl field and it is defined as follows:

- 0: Put PM (Packet Mode) tokens from PE in separate PM Token FIFO
- 1: Put PM tokens from PE in Axc Token FIFO to improve CPRI packet performance

This bit was added to deal with the CPRI packet starvation mentioned in the description for AD_ESCH_CFG.pri. Although it did not solve the issue, it was left as another possible performance tweak. When set to one, the non-AxC Multicore Navigator packet tokens (PM tokens) are sent to the same Token FIFO in the DB as the AxC Multicore Navigator packet tokens. Otherwise, the tokens are put into separate FIFOs.

The combination of AD_ESCH_CFG.pri = 1 and DB_EDB_CFG.pm_ctl =1 is the same as AD_ESCH_CFG.pri = 0 and DB_EDB_CFG.pm_ctl =1.

Programmable parameters for Multicore Navigator Egress Scheduler interface:

The base value of the 128 Multicore Navigator egress queues is programmable via **AD_ESCH_CFG.TXQ_QNUM**. AIF2 uses 128 Queues for transmitting from Queue number 512 to 640. **AD_ESCH_CFG.TXQ_QMGR** should be set to zero because the device supports only one Queue Manager.

7.11.3 Direct IO

The AD-DIO provides Direct I/O, non packet based DMA for WCDMA applications as well as for the Data Trace feature. The AD-DIO uses six DMA engines that are constructed specifically for WCDMA based transfers. The AD-DIO consists of three major components:

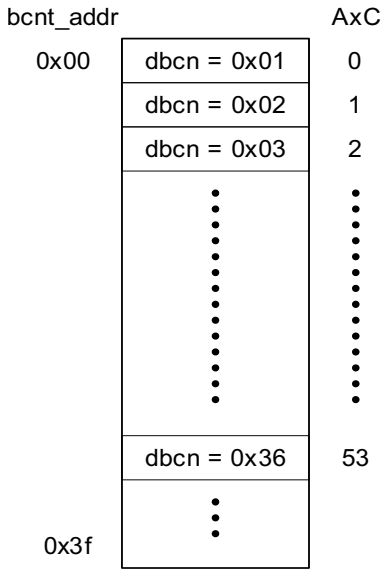
- Three Ingress DIO Engine for RAC, L2/RSA and DDR3
- Three Egress DIO Engine for TAC, L2/RSA and DDR3
- Data Trace DIO

For Ingress and Egress, the AD-DIO interfaces to the DB via the AD-SCH. For Data Trace, the AD-DIO interfaces directly to the DB.

Each of the six DMA engines has a Buffer Channel Number Table (DBCNT) LUT that the user programs with the Data Buffer Channel Number (DBCN) that corresponds to an AxC. This table is double buffered so that channels can be added or deleted. The user indicates to the DMA engine (bcn_table_sel), which table is to be active during the next frame.

In the example below, AxC0 is mapped to data buffer channel number 1, AxC1 is mapped to buffer channel number 2, etc.

Figure 7-29. Buffer Channel Number Table (RAC Example: w/ 54 AxCs)



When the DMA engine is triggered by the AT, it runs once through the entire list programmed into the DBCNT.

The buffer channel depth is either 128 or 256 bytes and is set via a configuration register in the DB. The AD-DIO will address the DB buffers in linear form and wrap at either 128 or 256 byte (that is, eight or 16 Quad Word) transfers based on how the DB is configured.

There are three engines for Ingress and three engines for Egress. Each Ingress engine may be assigned to any one of three destinations (RAC, L2/RSA or DDR3) simply by programming the appropriate base address. Each Ingress DBCNT may be programmed with the same source in the DB. In this way, data may be broadcast to several destinations from the same source.

Each set of three engines work one at a time. Each one completing a DBCNT worth of transfers before the next engine begins. There is a simple priority mechanism of engine 0 having the highest priority, then engine 1 and finally engine 2. This holds true for both Ingress and Egress. However, if engine 1 or 2 is triggered first, it will be allowed to run to completion before another engine begins. For example, engine 1 is triggered and begins, then engines 0 and 2 are triggered. In this case, engine 1 will run to completion, followed by engines 0 and 2 respectively.

Each DMA engine receives a dedicated trigger and frame boundary pulse from the AT (except the Data Trace engine). It is up to the user to program the AT correctly for the application.

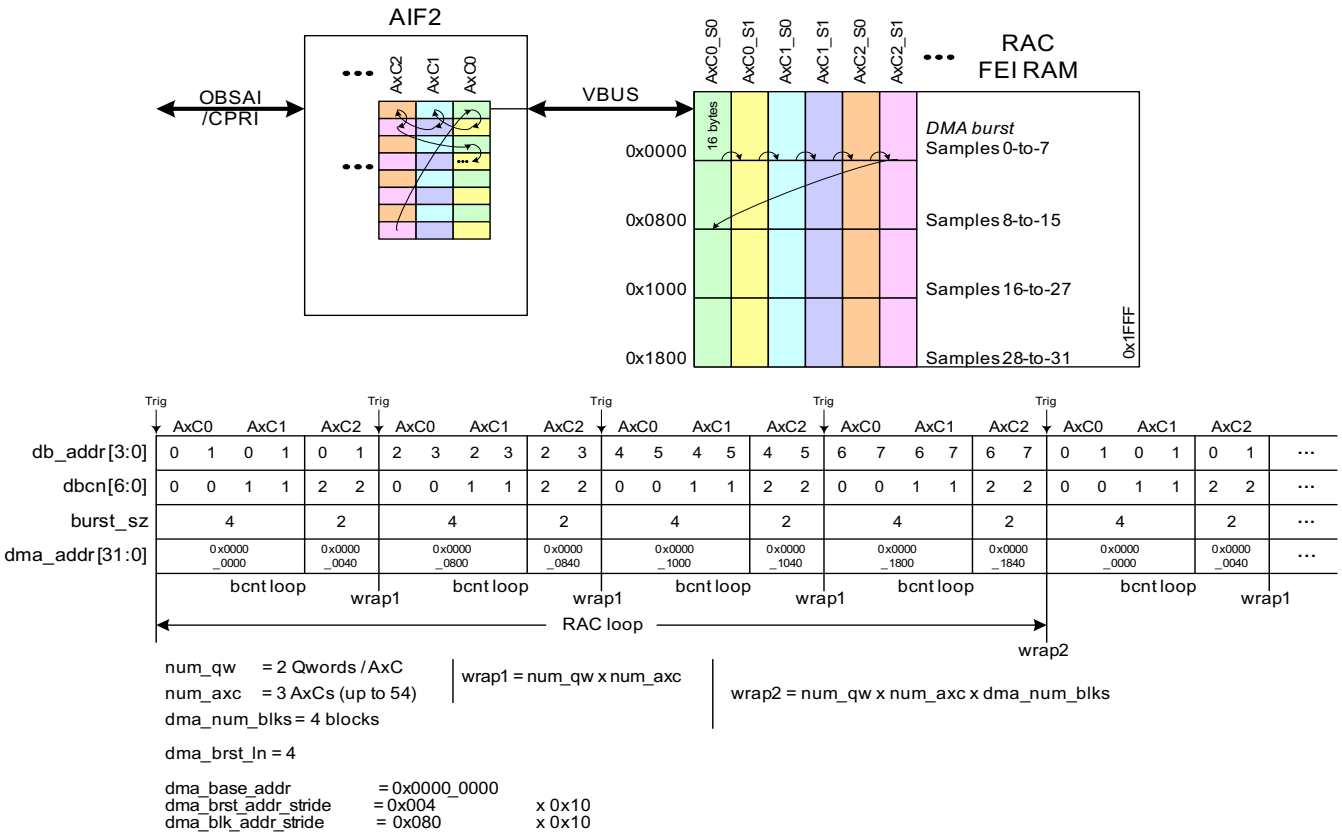
Each time an engine is triggered, it begins the DMA process, starting from the beginning of the DBCN Table and running through the table until completion. In the process of running through the DBCNT, a block of data is transferred. A data block is defined as the number of quad words per AxC times the number of AxCs (that is, **num_qw * num_axc**).

The engine does not trigger until it receives the first frame boundary pulse. When a frame boundary pulse is received, the engine uses the DBCNT LUT that is pointed to by the **bcn_table_sel** bit. It will continue to use this LUT until the next frame boundary pulse is received. If the **bcn_table_sel** bit has changed, it will use the other LUT. If the **bcn_table_sel** bit has not changed, it will continue to use the same LUT.

7.11.3.1 RAC Example

Figure 7-30 shows an example of how the DIO is programmed for RAC DMA. Trigger period is eight chips.

Figure 7-30. RAC Example

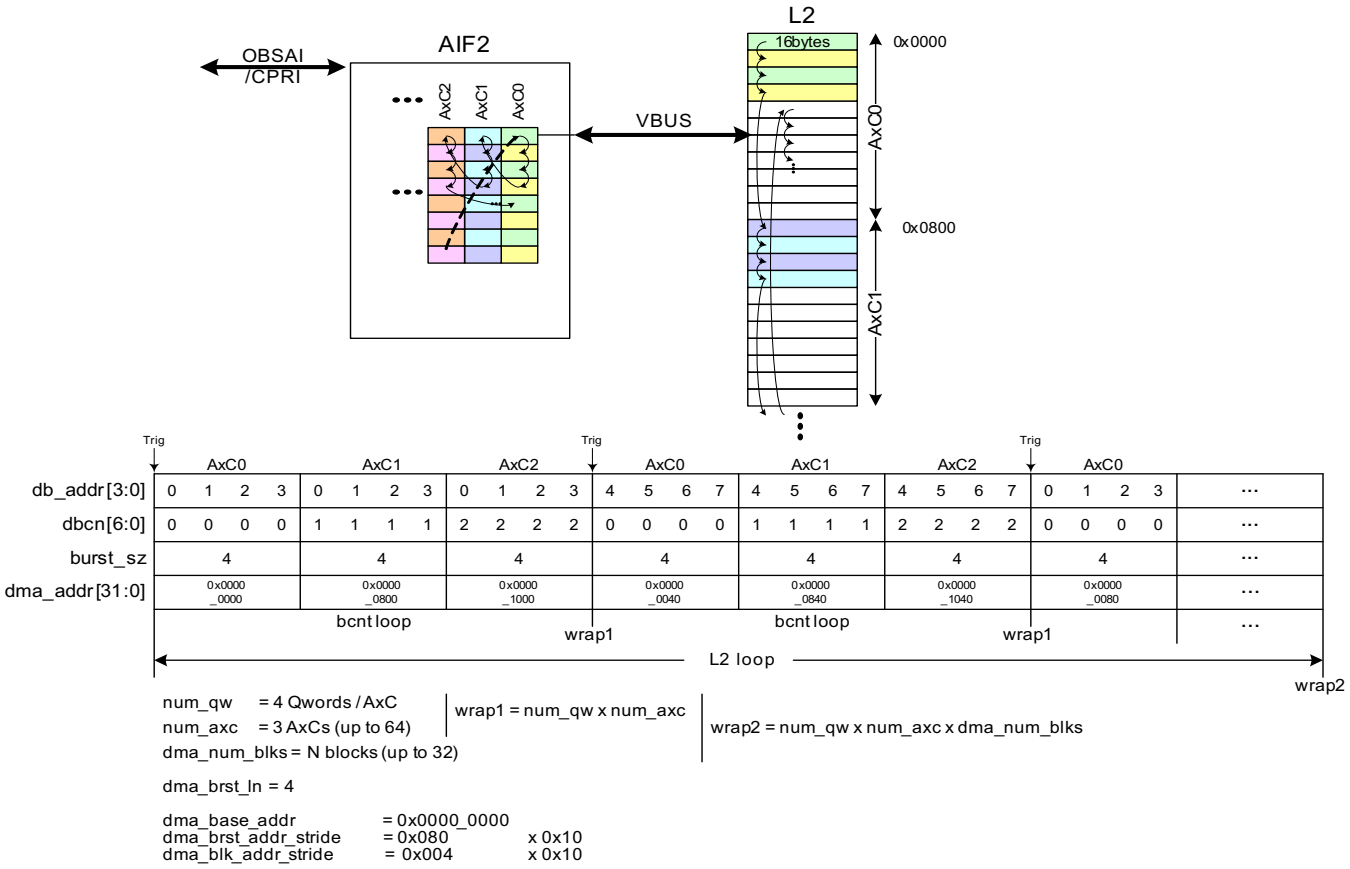


As the DMA engine traverses the DBCNT LUT, it is grouping data transfers into bursts of four Qwords. When it only has a burst of two left, it will only do a burst of two. The first burst will start at a VBUS address of **dma_base_addr**. After the first burst, the next burst will start at a VBUS address of **dma_base_addr + dma_brst_addr_stride** and so on until the DBCNT has been exhausted. After the next trigger, the DMA engine traverses the DBCNT again with starting VBUS address of **dma_base_addr + dma_blk_addr_stride**. The next burst will start at a VBUS address of **dma_base_addr + dma_brst_addr_stride + dma_brst_addr_stride** and so on until the DBCNT has been exhausted. The DMA engine will proceed in this manner until **dma_num_blks** have been transferred and then start at the beginning again.

7.11.3.2 L2 Example

Figure 7-31 shows an example of how the DIO is programmed for L2/RSA DMA. Trigger period is 16 chips.

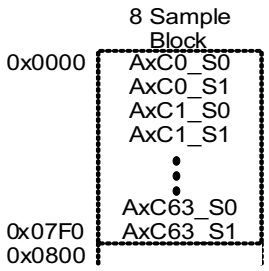
Figure 7-31. L2/RSA Example



7.11.3.3 DDR3 Example

DDR3 is used to store and later retrieve delayed data streams. Up to one frame plus one slot for 64 AxCs can be stored in this memory. There are 15 slots in a Frame, so the memory must be capable of storing up to 16 slots x 320 chips/slot = 40,960 chips of data for up to 64 AxCs.

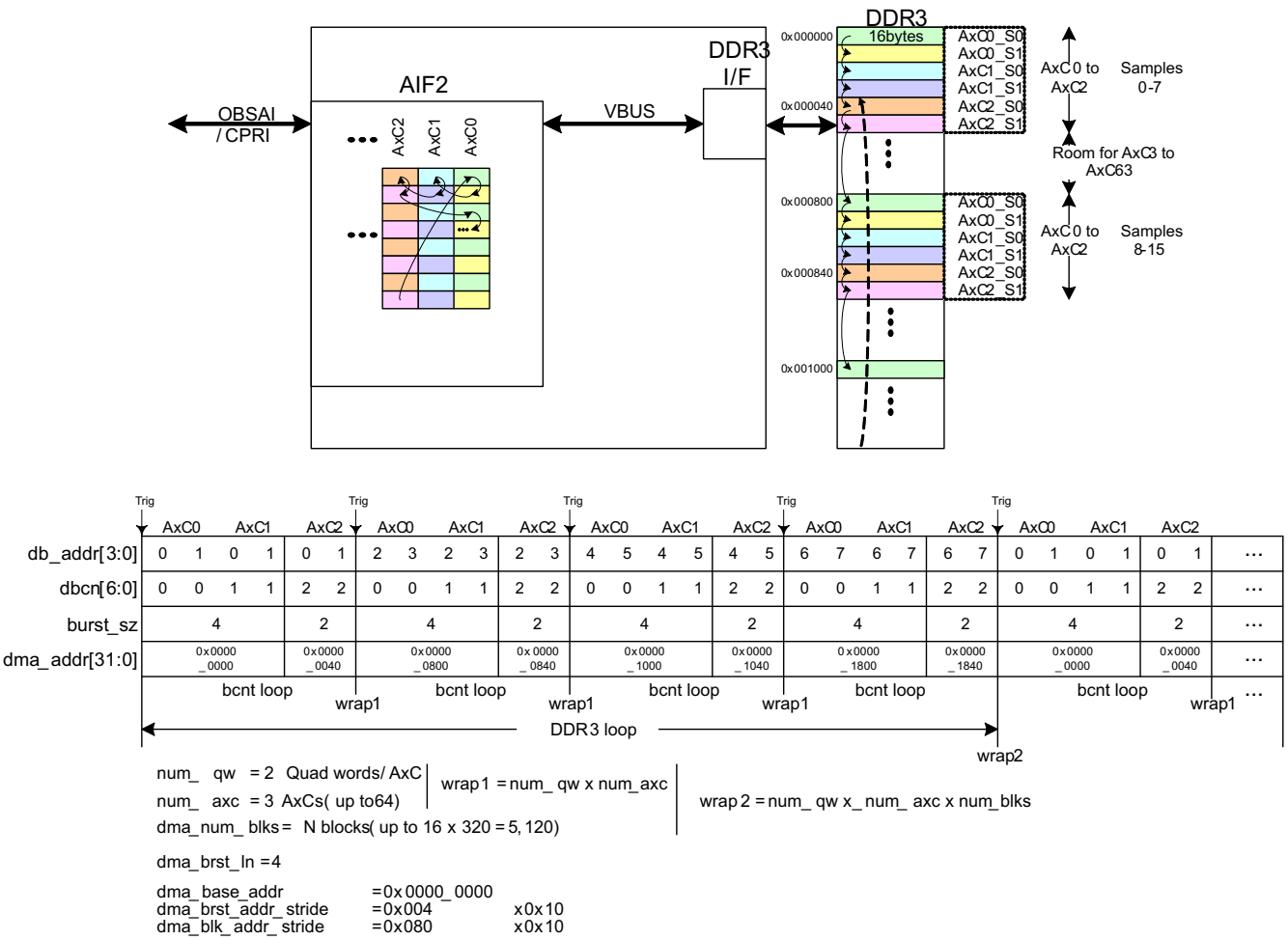
Figure 7-32. DDR3 Eight Chip Block of Data



An eight sample block consists of eight chips / 2 Qwords for up to 64 AxCs, or, 128 Qwords per block. At each trigger from the AT, an entire block of data will be transferred. There are up to 320 blocks per slot for up to 16 slots = 5,120 blocks.

Figure 7-33 shows an example of how the DIO is programmed for DDR3 DMA. The trigger period is eight or 16 chips.

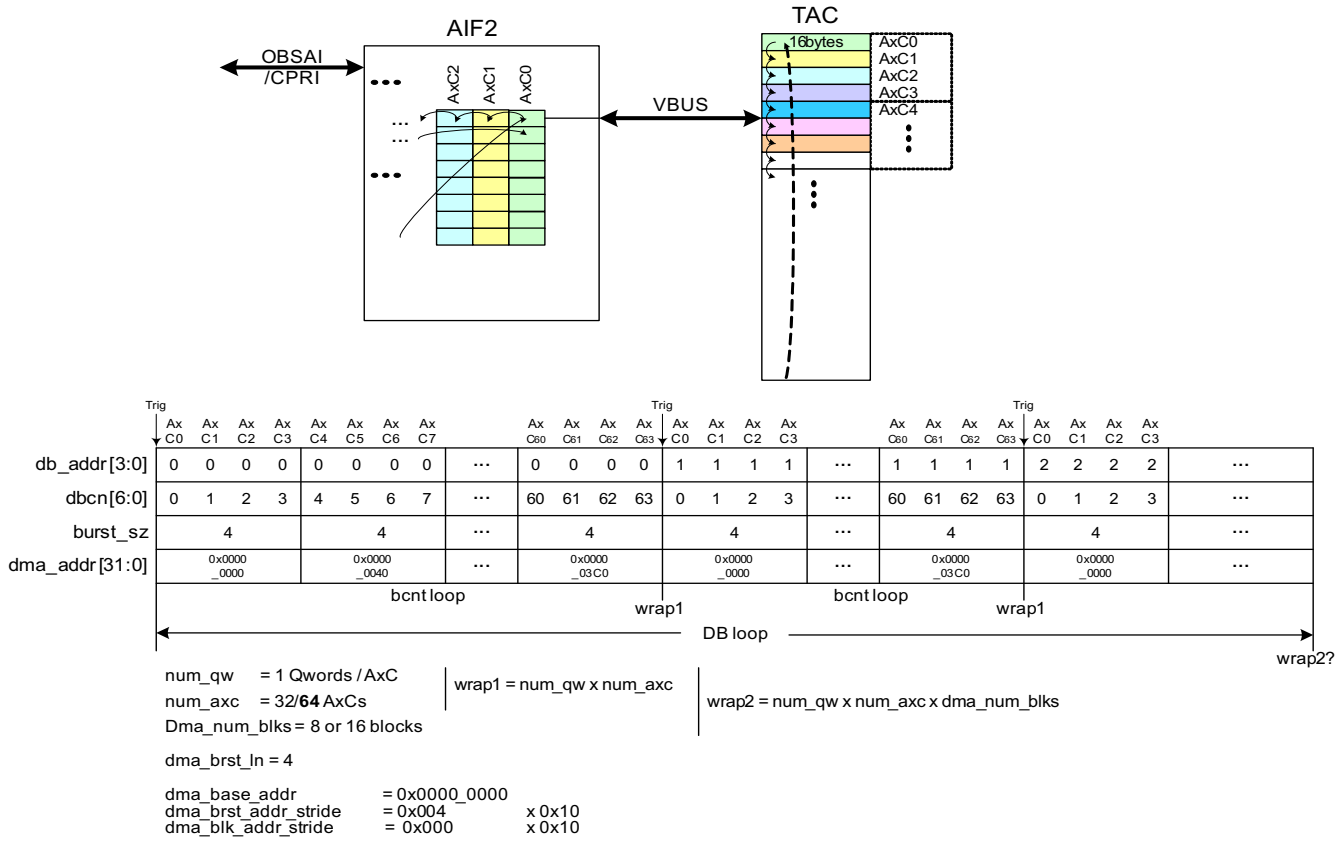
Figure 7-33. DDR3 Example



7.11.3.4 TAC Example

Figure 7-34 shows an example of how the DIO is programmed for TAC DMA. The trigger period is four chips.

Figure 7-34. TAC Example



7.11.3.5 Configuration Registers

bcn_table_sel - one bit

Selects which buffer channel number table for the DMA engine can be used at the next Frame Boundary.(support dynamic configuration)

- 0: Selects buffer channel number table 0
- 1: Selects buffer channel number table 1

num_qw - two bits

Sets the number of quad words per AxC.

- 00: 1 quad word per AxC
- 01: 2 quad words per AxC
- 10: 4 quad words per AxC
- 11: n/a

num_axc - six bits PKTDMA

Sets the number of AxCs (that is, set minus one value 0 to 63 for 1 to 64 AxCs).

dma_base_addr - 28 bits

Sets the VBUS source or destination base address (upper 28 bits of 32 bit data bus) this should be the global address in case of L2 memory.

dma_brst_addr_stride - 12 bits

Sets the DMA burst address stride (in multiples of 0x10 internally) After each DMA burst (64 bytes or less), the DMA address will be incremented by this amount. normally it is set to four QW (64 bytes) but if dma_brst_ln is 1 or 2 QW, then this value could be set to 1 or 2. This value could be very big if the a specific AxC data for whole frame should be placed in the same area like in L2 memory example.

dma_blk_addr_stride - 12 bits

Sets the DMA block address stride (in multiples of 0x10 internally) After each DMA block (every tigger time), the DMA address will be incremented by this amount. This value normally could be "Number of AxC * Number of QW per AxC" in case of L2 or DDR.

dma_num_blks - 13 bits

Set the number of data blocks to transfer before wrapping back to dma_base_addr. Block means the amount of data between triggers from AT. This value need to be set dependent on the destination memory block size.

- RAC: 32 or 64
- L2/RSA: Up to 32
- DDR3 (delayed stream): Up to 16 slots x 320 chips per slot = 5,200
- TAC: 8 or 16

dma_brst_ln - two bits

Sets the maximum DMA burst length. 4quad word is mostly recommended for burst length because the PKTDMA and AIF2 core transfer four quad words per transaction as a basic burst size except eop burst case.

- 00: 1 quad word transferred per burst
- 01: 2 quad words transferred per burst
- 10: 4 quad words transferred per burst
- 11: n/a

dma_ch_en - one bit

- 0: DMA channel disabled (cleared state)
- 1: DMA Channel enabled

rsa_en - 1bit

- 0: Egress DIO data is not RSA UL type
- 1: Egress DIO data is RSA UL (2QW) type

7.11.3.6 Data Trace DIO

The Data Trace DIO controls the transfer of data from two DB buffers to L2 or DDR3 memory. The data trace data consists of “raw” received data (16 bits) plus framing data (four bits). The “raw” received data is stored in one dedicated DB buffer and the framing data is stored in a second dedicated DB buffer. When the DB has four Qwords of receive data and/or one Qword of framing data, the DB signals to do a burst transfer.

The burst length is automatically set to four quad words for receive data and one quad word for framing data.

Data Trace receive data will be transferred to L2 or DDR3 starting at **dt_dma_rd_base_address** and up to **dt_dma_wrap** * 0x40 - 1 where data will begin overwriting old data starting at **dt_dma_rd_base_address** again.

Data Trace framing data will be transferred to L2 or DDR3 starting at **dt_dma_fm_base_address** and up to **dt_dma_wrap** * 0x10 - 1 where data will begin overwriting old data starting at **dt_dma_fm_base_address** again.

7.11.3.6.1 Data Trace Programming

dt_dma_rd_ch_en - one bit

DMA channel enable for data trace receive data DIO.

- 0: DMA channel disabled (cleared state)
- 1: DMA Channel enabled

dt_dma_fm_ch_en - one bit

DMA channel enable for data trace framing data DIO.

- 0: DMA channel disabled (cleared state)
- 1: DMA Channel enabled

dt_dma_rd_base_addr - 28 bits

Sets the destination VBUS base address (upper 28 bits of 32-bit data bus) for data trace receive data.

dt_dma_fm_base_address - 28 bits

Sets the destination VBUS base address (upper 28 bits of 32-bit data bus) for data trace framing data.

dt_dma_wrap - 18 bits

Sets the number of burst transfers before the destination address wraps back to the base address. that is, up to (16 bytes / Qword x four Qwords / burst) x 2^18 = 16 M bytes = 1.365 Frames of raw data at an 8x line rate.

7.12 AT (AIF2 Timer)

The antenna interface timer (AT) is meant to mark frame boundaries so that generated events can be synchronized with this time. Frame synchronization is flexible and currently known required framing boundaries are {1ms, 4ms, 5ms, 10ms, 60ms}.

There are two basic timer types used as a reference for PHY and radio timers (PHYT and RADT).

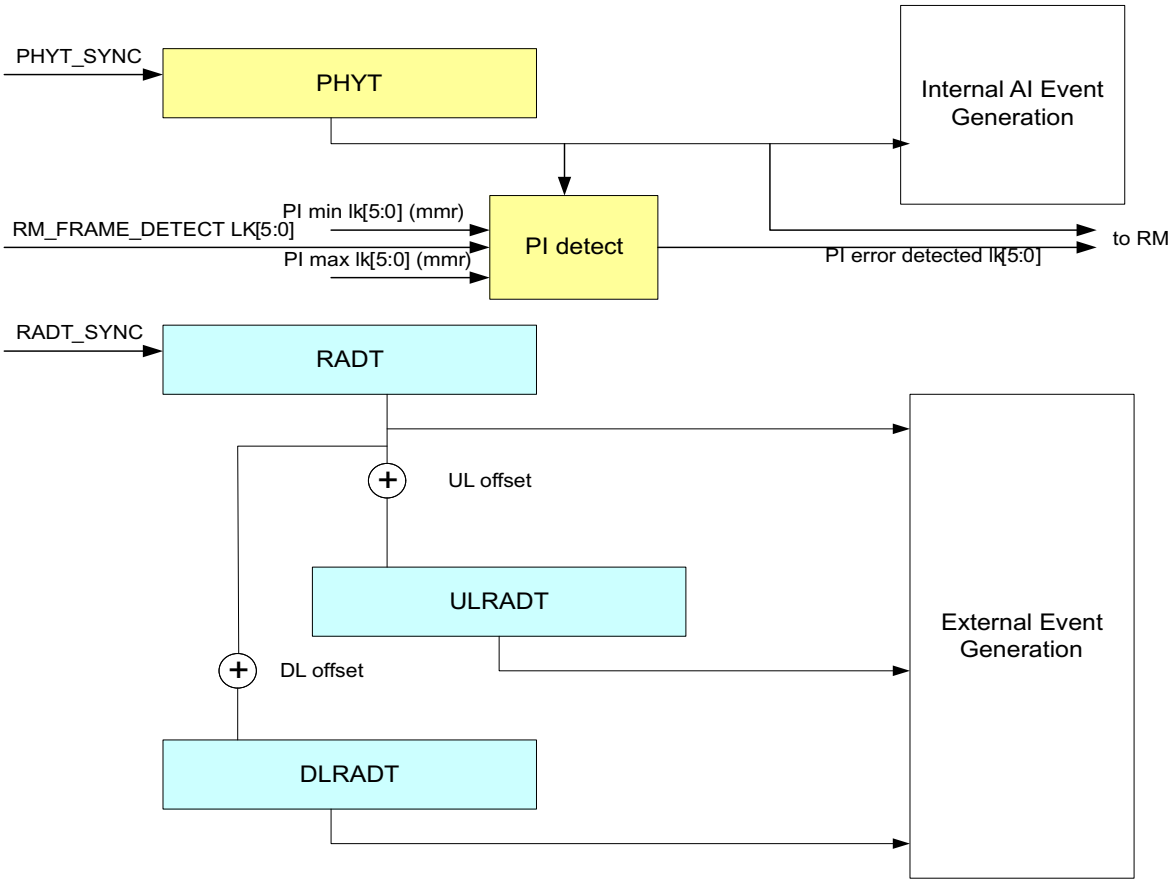
The PHY timer is used as a reference basis for link-based event generation. This timer is closely associated with timing of the received and transmitted link data traffic. It is used to direct link traffic and is used as a reference to set transmit Delta time and to check receive Pi time.

The radio timer RADT is used as a reference basis for radio standard event generation. This timer will be synchronized to a particular standard that is chosen.

Both timers are synchronized by a system reference from a clock control module (CCM).

The RADT timer will have two offset versions, ULRADT and DLRADT to be used to mark uplink and downlink radio standard time. These times are offset from the referenced RADT.

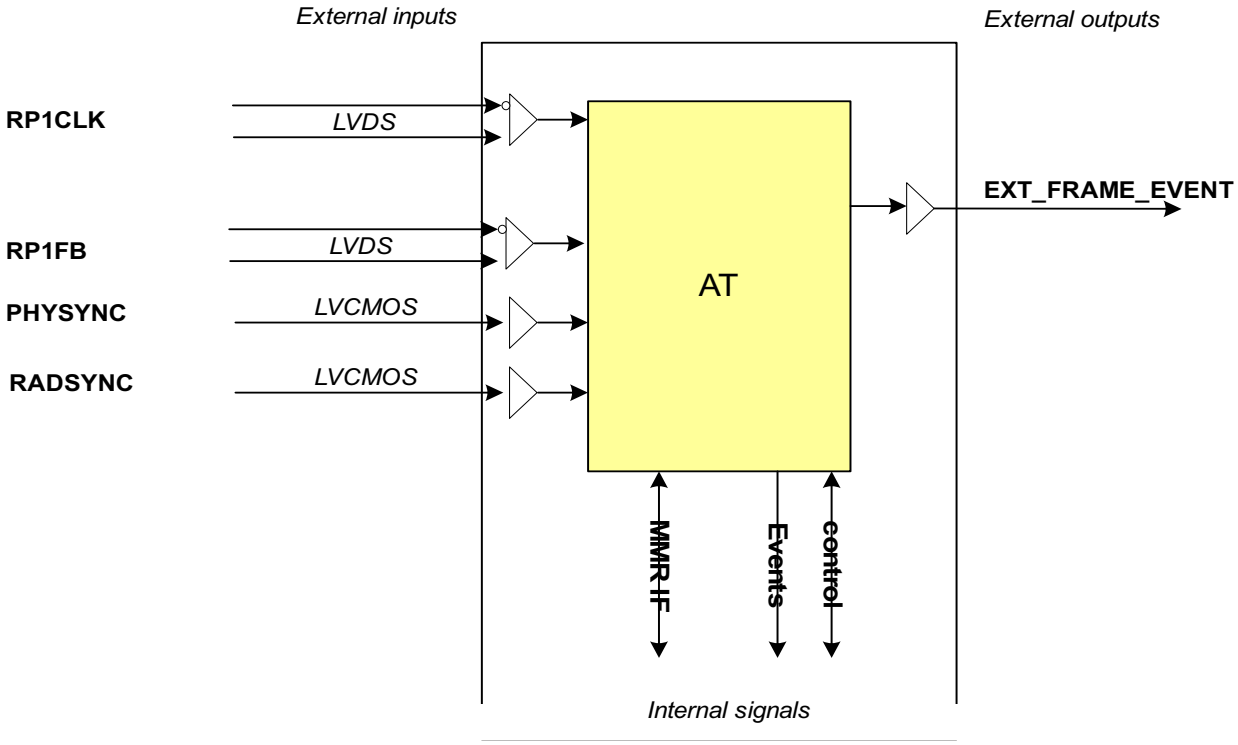
Figure 7-35. Timer Conceptual Diagram



The PHYT will be used for internal AIF2 events used in all links and DMA. The RADT timer and offset versions will be used by the system software to mark radio time. All timers are CPU readable.

Figure 7-36 shows the external IO. External IO goes to the chip boundary. There is one differential input clock for the OBSAI RP1 interface. There are three sync inputs, one of which will be selected for PHYTSYNC another will be selected for RADTSYNC, the third will be RP1 frame_burst data input. The internal interface for control/status MMR is the internal VBUS interface. One event, EXT_FRAME_EVENT, will be a chip output. This will typically be used for debug purpose.

Figure 7-36. AT Chip IO



There is a synchronization section that detects synchronization from an external source. This detected sync is used as a basis for the timers to start their count. There are two separate syncs, one for the PHYT and one for RADT.

7.12.1 AT Phy Timer (PHYT RP3 Timer)

The PHYT timer is used for marking frame boundaries. PHYT is used as references by the AIF2 for directing receive and transmit data traffic to/from the links. PHYT is used for detecting Pi error and generating Delta time. In OBSAI this would be considered to be the RP3 timer. The function of the timer is straightforward. There is a clock counter that runs at the basic AIF2 system clock rate of 307.2MHz for OBSAI, 245.76MHz for CPRI, and a frame counter that increments upon wrap of the clock counter.

The clock counter will count up to a full frame in OBSAI or CPRI mode. The terminal count for this counter is 25 bits wide and is programmable to count up to 4,193,304 clock ticks. A 10mS frame for example is 3,072,000 clock ticks (OBSAI). In case of short frame, it is 12,800 ticks for OBSAI (8 * link rate messages) and 32768 ticks for CPRI (2 hyper frames).

The frame counter is 40 bits wide. The initial value of the frame is loaded on the sync boundary.

There are two ways to load the frame number:

- Software sets initial value for full 40 bits (non-RP1 mode)
- RP1 frame burst loads a programmable amount, minimum eight bits, and software loads the rest

The frame count value may be initialized by software or could be updated by the AT_SYNC circuit in OBSAI RP1 mode upon receiving a PHYT_SYNC pulse. In RP1 mode the frame number may update the lower eight bits of the frame counter up to the full 40 bits of the frame counter upon receiving a RP3 type frame burst. These options are selectable in a control1 register. If the LSBs are to be updated by the RP1 interface, the upper bits can be software programmed.

The Phy Timer sync select option is shown in Table 7-23.

Table 7-23. Phy Timer Sync Selection

Selection	Sync Source	PHYTSYNC source Description
0	RP1_PHYT_SYNC	From RP1 interface FCB detection
1	PHYTSYNC	External pin
2	SW_PHYT_SYNC	From MMR
3	RM_ SYNC	From selected RM link Frame detect

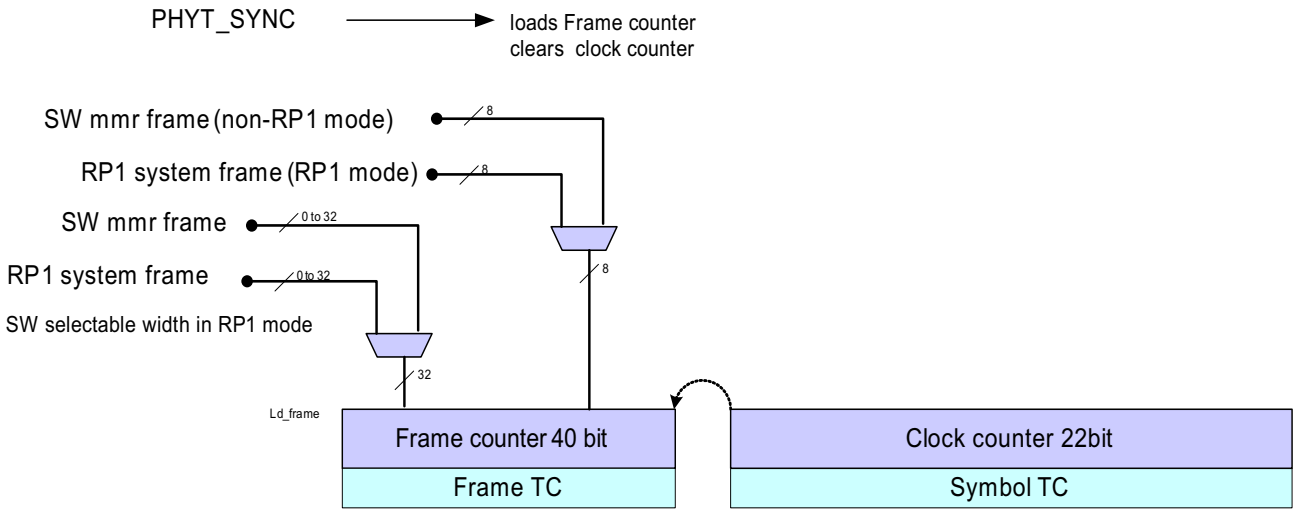
RM Link synchronization detection is used when Framesync is in non-RP1 mode. The synchronization source is received from the RM module. This is derived from the frame synchronization boundary received from a selected link. When detected, a single internal sync pulse is sent to the selected armed timer and starts the timer count and event generation associated with that timer.

- The sync source for PHYT would be RM_AT_SYNC.
- The sync source for RADT would be RM_AT _SYNC.

Terminal counts for the frame counter and symbol counter are initialized by software. PHYT counters are readable by software. The PHYT clock counter has 22 bits but an extra two LSBs that are fixed at 00 are added to help conform to system time standards.

The PHYT receives it's synchronization from the AT_SYNC module as a single-clock high true pulse PHYT_SYNC. The PHYT will only start counting if it is first armed by software and then receives a PHYT_SYNC pulse.

Figure 7-37. PHYT Timer

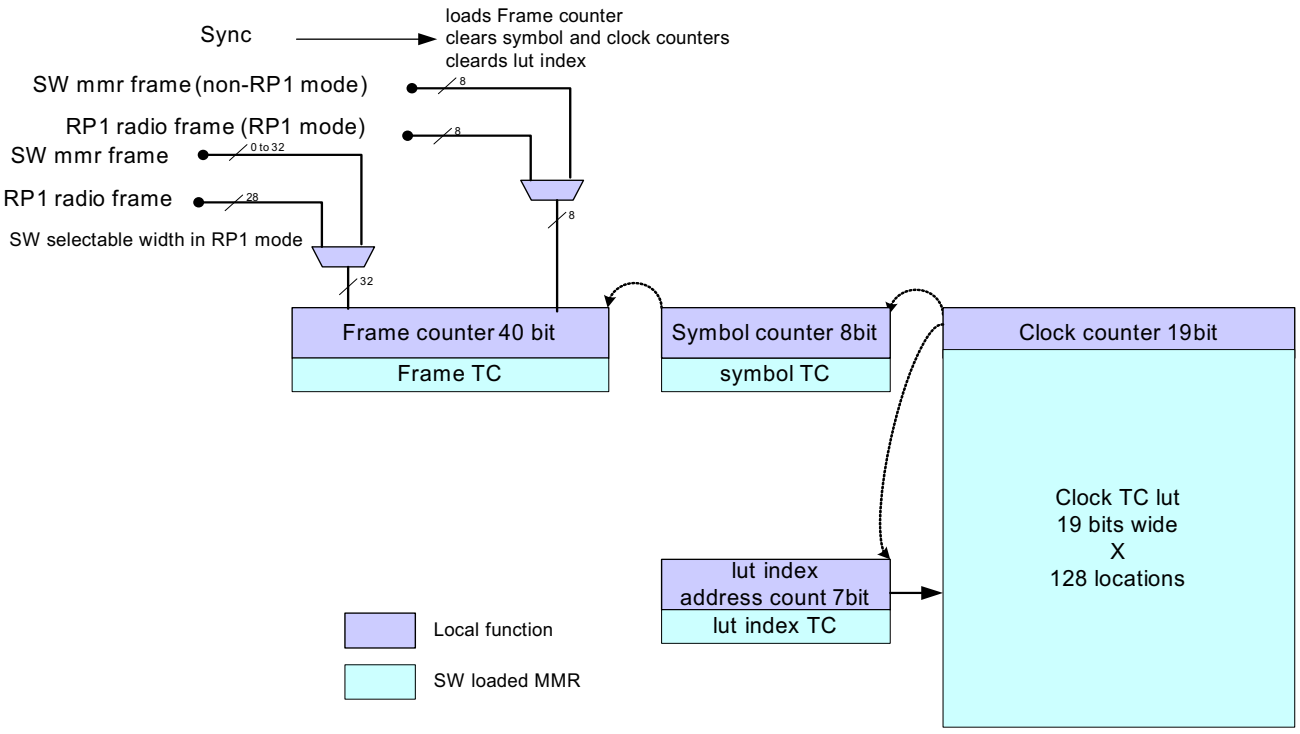


7.12.2 AT Radio Timer (RADT RP1 Timer)

General Radio Timer will have the ability to count symbols and frames of varying sample counts. Symbol or frame boundary will be used to drive event generators. These events will be used by the system external to AIF2. These events will typically interrupt CPUs but may be used by other system components.

Timer symbol counting will be accomplished with the use of a lookup table (LUT) programmed to represent the radio standard time. This allows for timing a regular pattern of symbols of variable size as in some TDM standards. The basic counter setup is as diagrammed in Figure 7-38.

Figure 7-38. RADT Basic Timer



There is a fine-grain reference timer clocked off of the TX byte clock. This time is converted to its selected standard through the use of the Clock TC lut. The lut is populated (programmed) with terminal counts that represent all of the symbol terminal counts within the desired standard's frame or group of frames. A separate symbol lut index counter is used to select the clock counter terminal count value for that particular symbol. It will increment whenever it sees a symbol boundary and will wrap on the number of symbols per frame counted. The wrap value is programmed per standard. Typically, the wrap value of the lut index is equal to the wrap value of the symbol TC. An exception is in LTE, which has a repeating pattern or either six or seven symbols within a group of 120 or 140 symbols (this will be explained in the section on AT Event Generation).

The RADT timer is started when it is both armed and it receives a RADT_SYNC. The RADT_SYNC can be derived from selectable sources. Table 7-24 shows the options.

Table 7-24. Rad Timer Sync Selection

Selection	Sync Source	RADTSYNC source Description
0	RP1_PHYRADT_SYNC	From RP1 interface FCB detection
1	RADTSYNC	External pin
2	SW_RAD_SYNC	From MMR
3	RM_AT _ SYNC	From selected RM link Frame detect
4	PHYT CMP_SYNC	From PHYT value compare

Phy and Rad timer has separate software sync for software testing so it is hard to make perfect synchronization when triggering both sync. PHYTCMP_SYNC option solves this problem. user only trigger Phy software sync and Rad sync is automatically triggered by comparing phy sync signal.

When PHYT compare value is selected, the value of the PHYT FRAME count and CLOCK count is compared against a programmed value. When this value is reached in the PHYT, the RADT sync is issued.

ULRADT timer is programmed with a delay from the reference RADT. This is done by programming a different initial value for the clock count, symbol count, lut index count and one bit (ULFCB_minusone) for the frame. It is started when it is both armed and it receives the RADT_SYNC. Offset must be less than one frame, positive or negative. UL FCB_minusone bit should be set only when the time difference between ULRADT and RADT is more than half frame time not to increase the frame count.

In RP1 mode, AT will receive a supported radio standard FCB and capture the Frame number. This 40-bit frame number may be loaded into the RADT frame counter 40 bits or it could be loaded into the lower eight to 40 bits with the upper bits initialized by software. This software load would typically happen after the RP1 FCB is received by reading the FCB value, then writing to the RADT initialization register with the modified value.

If the RADT is to be updated by software, then ULRADT and DLRADT must be updated as well. At the same that the RP1 FCB is received (RADT_SYNC), the ULRADT Frame counter will be loaded with this same width of the RP1 Frame number value or the Frame number -1, depending on the ULFCB_minusone bit in the UL control register, at_ulradt_init_lsbs bit 31. Only the selected RP1 frame width will be loaded into the RADT and only the same RP1 frame width will be updated in UL RADT depending on the UL FCB_minusone bit. The unselected Frame MSBs will be unchanged upon RADT_SYNC.

The DLRADT timer operates the same way as ULRADT but must have the timer initial values programmed for its unique offset from RADT, including DLFCB_minusone.

There is an additional GSM timer to the base RADT timer that is not included in ULRADT or DLRADT. This marks T1, T2 and T3 frame times. These counters count GSM frames with a duration of 1/13 th of a 60 ms period. Modulo count for the three timers is as follows:

- T1 = modulo 2048
- T2 = modulo 26
- T3 = modulo 51

The T2, T2 and T3 counters do not get updated or initialized by the RP1 interface FCB. Software must initialize the T1, T2 and T3 counters based on the current FCB received. The software application must calculate T1,T2 and T3 based on a received FCB frame burst and then load these T counters.

T2 and T3 both count GSM frames. T1 counts every time both of these counters wrap. Max count for T1 is 2047 before it wraps to 0.

There are three additional special counters for WCDMA value.

Radt WCDMA value register shows chip, slot and frame value when Rad timer is used for WCDMA. UL and DL timer also has this register and radt wcdma div register let us set clock divider terminal count and this counter divides dual byte clock (307.2 MHz or 245.76 MHz) down to 3.84 MHz chip rate for WCDMA counter. Normally it is set to 79 for OBSAI and 63 for CPRI to divide dual byte clock by 80 or 64, but if user wants to change the dual byte clock higher or lower, this value also could be changed to get correct 3.84 MHz chip rate.

AT_Captradt register shows the captured value for radio clock, symbol, and frame value that count upon a Phyt frame boundary. Users may check the time offset between Rad timer and Phy timer.

7.12.3 AT OBSAI RP1 Synchronization

Detection of synchronization in RP1 mode is more involved than simple synchronization.

OBSAI RP1 specifies that the CCM (Clock and Control Module) provides a 30.72 MHz System Clock and periodically sends synchronization bursts to the BaseBand Modules. These synchronization bursts are received by Framesync on the BB module where UMTS timing information is extracted. The RP3 and System Frame Numbers, Frame boundary timing and Time of Day information is passed in the synchronization bursts.

The OBSAI synchronization burst is serially transmitted over a single differential input (differential signaling) that is clocked in through the differential SCLK. Each bit of the serial transfer is held for eight System Clock (SCLK) periods; This is approximately 260ns. For fields with more than a single bit, the least significant bit (LSB) is sent first.

The first field is the “Start” bit, which marks the beginning of the synchronization burst. The 8-bit “Type” field follows and identifies the type of information contained in the synchronization burst payload. The 64-bit “Information” or Payload field contains the relevant data (either frame number, or time of day). The CRC field is used for data integrity and the “End” field terminates the synchronization burst packet.

Figure 7-39. RP1 Synchronization Burst Format

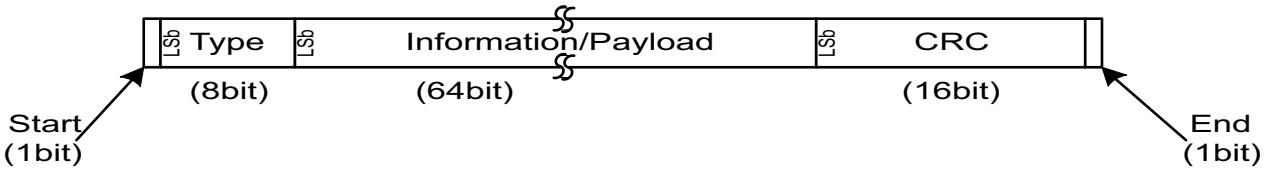
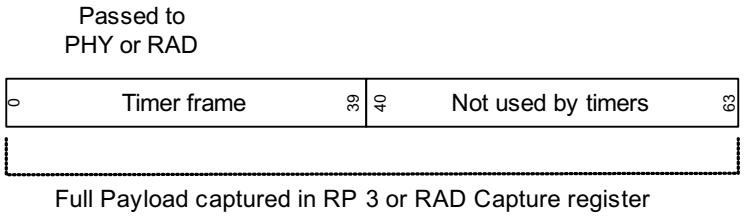


Table 7-25. RP1 Type Field Definition

Type	Supported	Value
Not Used	N/A	00h
RP3 Bus (FDD) Frame Number	Yes	01h
WCDMA/FDD Frame Number	Yes	02h
GSM/Edge1 Frame Number	Yes	03h
GSM/Edge2 Frame Number	Yes	04h
GSM/Edge3 Frame Number	Yes	05h
WCDMA/TDD Frame Number	Yes	06h
PKTDMA 2000 Frame Number	Yes	07h
Time of Day	Yes	08h
Reserved	N/A	09 - 7Fh
802.16 Frame Number, 2ms Frame	Yes	80h
802.16 Frame Number, 2.5ms Frame		81h
802.16 Frame Number, 4ms Frame	Yes	82h
802.16 Frame Number, 5ms Frame	Yes	83h
802.16 Frame Number, 8ms Frame	Yes	84h
802.16 Frame Number, 10ms Frame	Yes	85h
802.16 Frame Number, 12.5ms Frame		86h
802.16 Frame Number, 20ms Frame		87h
3GPP LTE Frame Number	Yes	88h

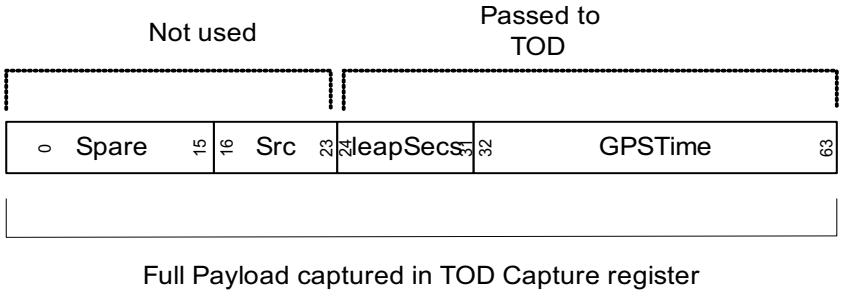
The Frame Synchronization Module will support the RP1 interface type WCDMA/FDD. The payload is used as follows for the PHY or RAD timers.

Figure 7-40. Payload Use for Type RP3 and WCDMA FDD



The payload is used as follows for the Time of Day timer.

Figure 7-41. Payload Use for Type Time of Day



When synchronization has already been established and a new sync arrives while the PHYT and RADT timers are counting, the AT will make a determination if this new sync is on the correct boundary. Because the sync input is retimed with the internal 307.2MHz system clock, there will be some uncertainty. The amount of uncertainty that the user is willing to tolerate may be programmed into the at_control1 register “sync_smpl_window” field. The number in this field represents the ± window minus1. For example, if the value in sync_smpl_window is 0, there will be an allowed ± one clock cycle of uncertainty (default).

If the programmed value is 5, the uncertainty will be ± five clock cycles. This error window checking is based on the original sync that the timers are based off of or on the resync sync if the auto_resync bit is set. Error checking for the phyt_sync alignment will start after the first frame boundary. If the PHYT has a programmed initial offset, error checking will start once the first frame boundary is detected. Error checking for the radt_sync alignment will start after the third full frame boundary.

At minimum, this alignment error may set an error type interrupt if that error bit is unmasked. If the at_control1 “auto_resync” bit is set, the affected timer will resynchronize on the new frame boundary, loading in the new current frame number into the frame field of the timer. Because the RP1 frame load width is programmable from a minimum of eight bits to a maximum of 40 bits, only the RADT selected Frame lsbs will get loaded. The msbs will be unchanged. The DLRADT and ULRADT will have their Frame lsbs updated with either the RP1 value or RP1 value -1. Again, only the selected Frame lsbs will be updated and the Frame msbs will be unchanged. If AT is not in RP1 mode and in auto-resync, the initial Frame value will be reloaded into the frame counter upon rad_sync error for RADT, ULRADT and DLRADT. For phy_sync error in non-RP1 and auto-resync modes, the initial frame value will be loaded.

Also, the clock count, symbol count, lut index will all go to their programmed initial value. In WCDMA this would all be 0 for the RADT and will be the programmed initial values for the ULRADT and DLRADT.

Events will immediately adjust to the new frame or symbol boundary.

RP1 startup and usage

1. Control register 1 is used to select parameters for RP1 usage and must be programmed first. This register sets up the following:
   (a) Timer synchronization source
   (b) RP1 mode
   (c) Auto-resync
   (d) RP1 CRC parameters
   (e) RP1 Frame load size for PHYT and RADT (default is 8)
2. At_rp1_type register needs to be programmed with the desired radio standard type.
3. Program terminal counts for all timers, including radio timer LUTs.
4. Program ULRADT and DLRADT initial values for clock count, lut index, symbol count.
5. Program the ULFCB_minusone and DLFCB_minusone bits if clock and symbol init value is ahead of Rad timer's init value to align the frame value.
6. Program the event timers and event input strobe selects for internal and external events.
7. Program the PI windows.
8. Arm the AT by setting control register 2 arm_timer bit.

Control register 1 has phy_syncsel and rad_syncsel field and the user normally can use RP1 interface synchronization option but it might show two clock delay between phy sync and rad sync. That delay comes from the HW and it could make problematic situation if user doesn't want any minor time delay. In that case, compared Phyt value sync option can be used for rad_syncsel to remove such problem. If compared Phyt value sync option is selected, user may have two options for initialization.

Option1: Set **at_phyt_cmp_radsync** value to **phyt_clock_tc -1**

Phyt and Radt will be fully aligned at second phy frame boundary position. When the phyt has one more cycle to go, radt is kicked off, such that the next cycle, phyt_clkcnt and radt_clkcnt will be aligned to zero. In this case, Phyt will be started one frame ahead than Radt and it could affect all other AIF2 timing parameters.

Option2: Set **radt_clkcnt_init** value to **2** and leave the **compare phyt value** to **zero**

Phyt and Radt will be fully aligned from first frame boundary position. Two clock delay between phyt and radt will make those timers fully aligned

For both cases, The AT decides to capture the RADT counters when it detects a PHYT frame boundary, (that is, phyt counter is at terminal count, therefore, radt counters are also at terminal count). So at_captrad counter may show terminal count value for the clock count as an initial value instead of zero.

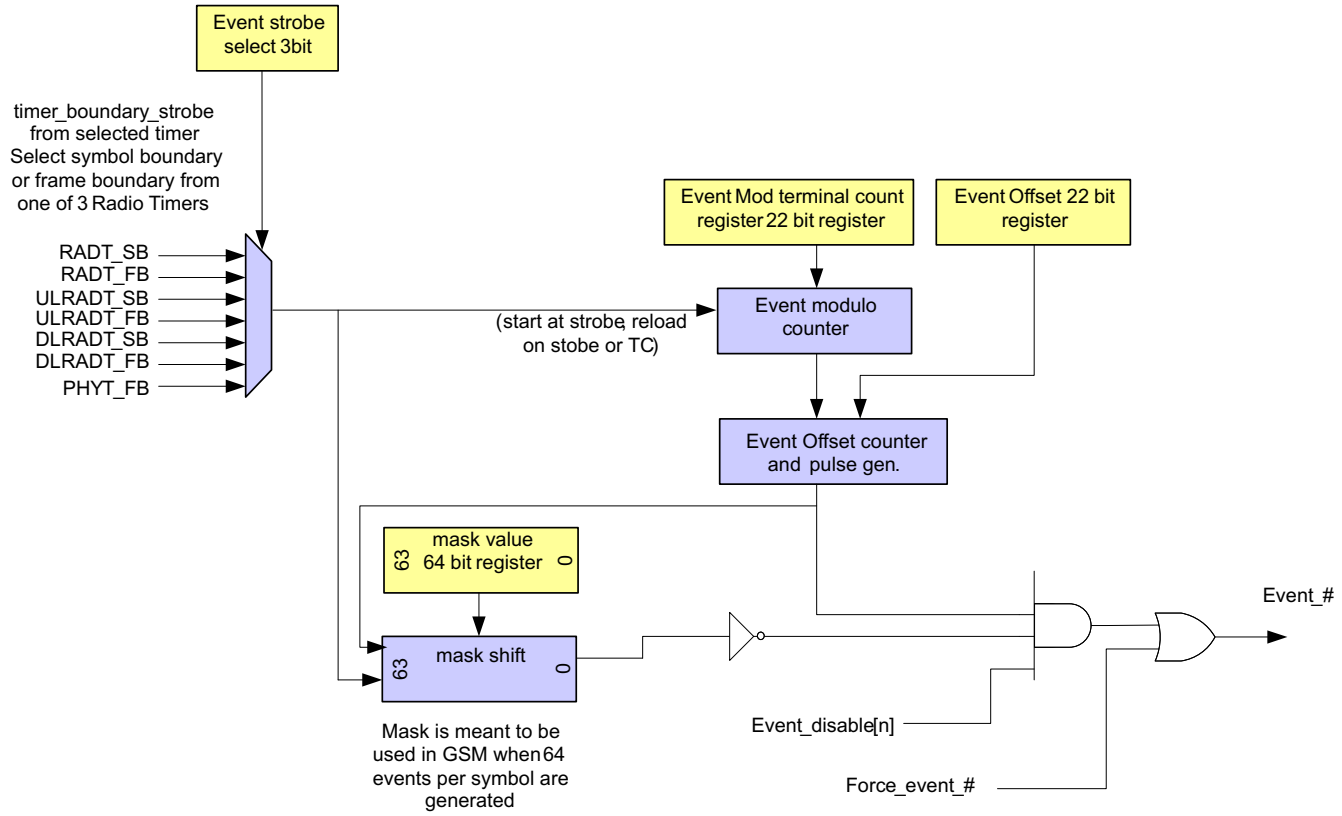
7.12.4 AT Event Generation

There will be **11** events used by the system outside of AT and within device and one event external to the device. These events will be referenced to the radio timers (RADT timers). Events for internal AT use are based on the PHYT timer.

Each event generator for the 11 events used within the device is working based on an event offset, event modulo and event strobe selection. These values are set in MMR. There is a boundary (symbol or frame) detect signal from its selected timer (RADT, ULRADT, DLRADT). This boundary is typically a symbol boundary, but may also be a frame boundary and the user can choose this by strobe selection field for each event.

Figure 7-42 shows how AT event generator works.

Figure 7-42. Event Generator



Detection of the boundary will start the event modulo timer inside the AT. Events will occur upon compare of the modulo timer with the Event offset count, provided that the particular event is not masked. There is a 64 event mask register that is set to mask any combination of 64 events in a symbol, provided that max 64 events per symbol are programmed in the event that generated from modulo and offset counter. This is meant to be used in GSM and it should be set to all 1's for all other radio standards.

at_event_offset[0] ~ [10]: set 22 bit event offset and 3 bit strobe selection

at_event_mod_tc[0] ~ [10]: set 22 bit event modulo terminal count (set n-1)

at_event_mask[0] ~ [10]: set 64 bit mask value for GSM (lsb for lower 32 bit and msb for higher 32 bit)

In case of TDD system, user may change UL, DL ratio by using this 64 bit mask. If the number of events per frame is 64 or less and user program an event to be strobed on a frame boundary, user can program that event to be either UL or DL but would need to program the mask on the fly before the frame boundary that requires the change in ratio. If the number of event is bigger than 64, the user can not use this mask for TDD.

Event offset and Event modulo TC should be less than the event Symbol period (or Frame period) and event offset should be smaller than Event modulo TC value. If user selects modulo TC to 319 byte clock, the max event offset should be smaller than 320 and that is why AIF2 has DIO frame event to set big amount of event offset.

Event modulo minimum is mod 8 and maximum is the size of boundary detect signal (Symbol or Frame). This will allow for crossing synchronization boundaries into the VBUS clock domain which is CPU clock/3. If user set bigger modulo than these boundary detect signal, it will be gently ignored and event will occur like it is set to legal maximum modulo value (Only Symbol or Frame event could be seen).

Because all of the events are counter-based, each event can have the flexibility to have a modulus that is not necessarily a power of 2 but Clock TC plus one value should be dividable by modulo value to make periodic modulo events between symbols or frames.

Events may be disabled or enabled on the fly while events are being generated. Disabling an event will cause the event to be removed on the next available selected timer frame boundary (frame or symbol). An event that is enabled during event generation will start generation upon its programmed offset after the next detected boundary.

Event enable/disable register is mainly used to start or stop event generation and halt timer function is used to stop Phy and Rad timer but the halt timer is mainly used for customer software debugging and not for main event disable mechanism.

Events can be forced in software. This feature is used for system debug or DB debug.

AT module has two external sync input for phy and rad timer. Periodical sync or one time sync could be used to trigger each timer but should be careful when periodical sync is used not to have delay or jitter between sync pulses. This delay could start resynchronization process and current clock, symbol or frame counter value could be broken.

In case of software debug sync, both phy and rad sync will be triggered at the same time. AT sync register should be set only once when the whole user program is ready. Multiple set of this register can cause the resynchronization of each timer.

The external system event, EXT_FRAME_EVENT, is a chip output event based on the RADT Frame boundary. The requirement for this event is to generate a frame pulse that is a minimum of eight byte clock cycles long.

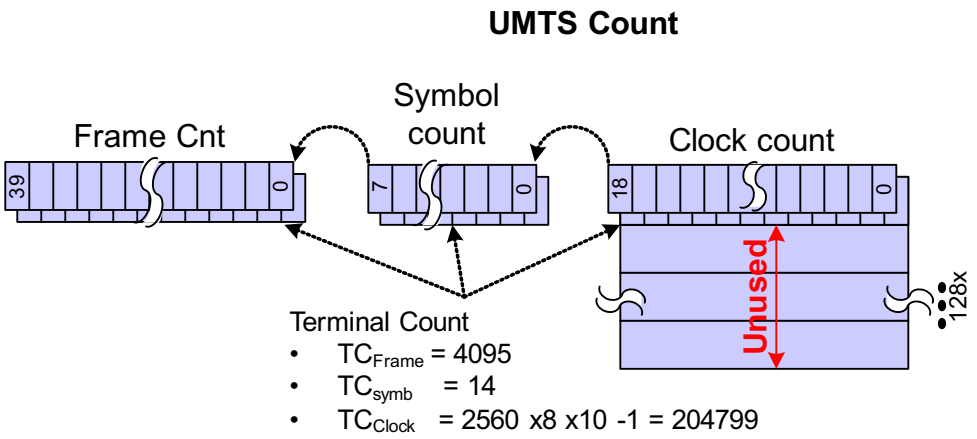
Table 7-26. Use of Timer Fields for Different Radio Standards

Radio Standard	Frame_Cnt	Symbol_Cnt	Clock_Count	Modulo
WCDMA	10ms Frames	Timeslots	Clocks per Slot	4chip, 8chip
LTE	10ms Frames	1ms subFrames	Clocks per subFrame	Symbol
WiMax (TDD/FDD)	Frames	Symbol Count	Clocks per Symbol	Sub symbol
TD-SCDMA (TDD)	Frames	Slot Count	Clocks per Slot	432 chip **(half slot)**
GSM	60ms	Timeslots per 60ms	Clocks per Timeslot	Sub Slot (total 64)

7.12.5 AT WCDMA Counter

The RAD timers will be programmed as shown in Figure 7-43 for WCDMA.

Figure 7-43. WCDMA RAD Timer Setup



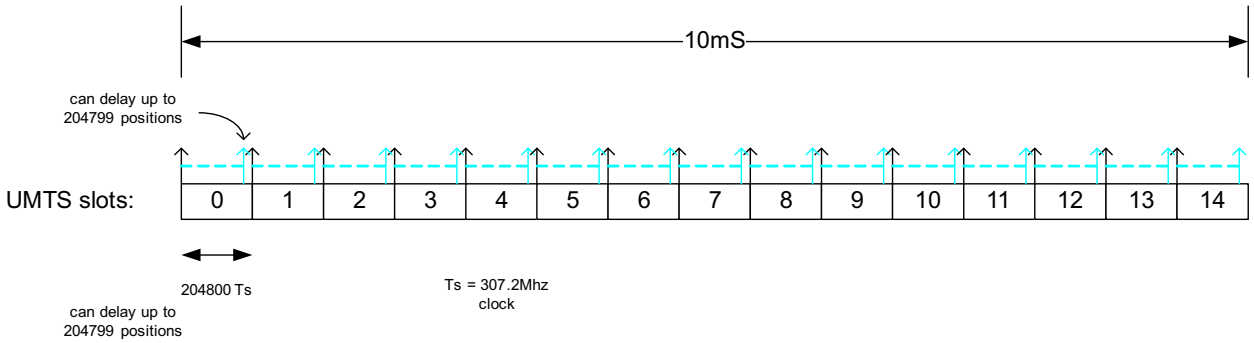
The LUT index will always point to one location. This location will have the terminal count value that the clock counter will count to, 204799 (OBSAI). The clock counter will count 2560 chips.

Symbol count TC will be 14, allowing for 15 symbols per frame.

The frame counter will count up to 4095 then return to zero.

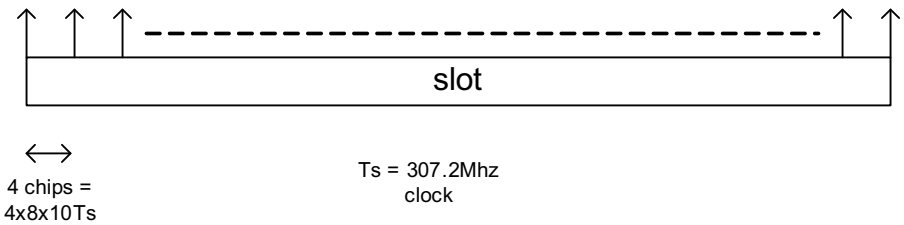
RADT events will be triggered by either a symbol boundary or frame boundary. Figure 7-44 shows one event per slot.

Figure 7-44. WCDMA Event Per Slot (OBSAI)



Once triggered, there could be up to 640 events per symbol (UMTS slot), representing one event for every four UMTS chips. Figure 7-45 shows one event for every four chips.

Figure 7-45. WCDMA Event Every Four Chips (OBSAI)



Event 8, 9 and 10 is assigned to special purposes. Event 9 is directly connected to the RAC_A (Receive Accelerator A), Event 10 is directly connected to the RAC_B and makes 32 chip periodic signals to start the processing and Event 8 is directly connected to the TAC (Transmit Accelerator) and supplies four chip periodic trigger to start processing. User can not use these three events for general purpose for application.

There are three WCDMA value registers (Radt, Ul radt, Dl radt) and these registers show the chip, slot and frame lsb value for each timer. These values could be read by TAC or RAC application to feed correct timestamp value

For WCDMA, there are special six internal Direct IO events for Ingress and egress trigger. User can use this to generate 4chip (for DL) or eight chip (for UL) trigger to start transfer data through Direct IO mechanism. It is also using RAD,ULRAD, DLRAD init registers and terminal count like external events, but has its own offset and modulo registers. Please see section 6.2.1 for more details. Modulo and offset is using byte clock, so 80(OBSAI) or 64(CPRI) clock time is same to WCDMA 1 chip time (260 ns). This also could be used to calculate Delta, Pi, PE1, PE2 offset and modulo value. See Section 4.5 for more information about these Phy level internal events.

7.12.6 AT LTE Counter

Figure 7-46. LTE RAD Timer Setup

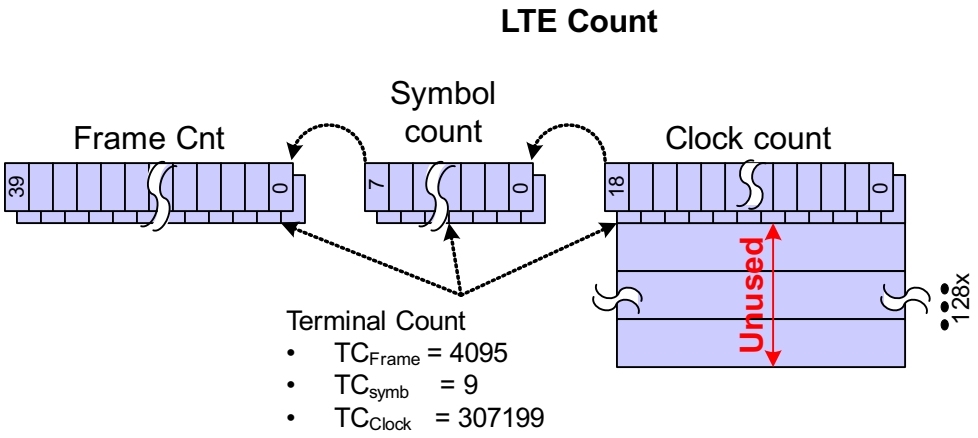
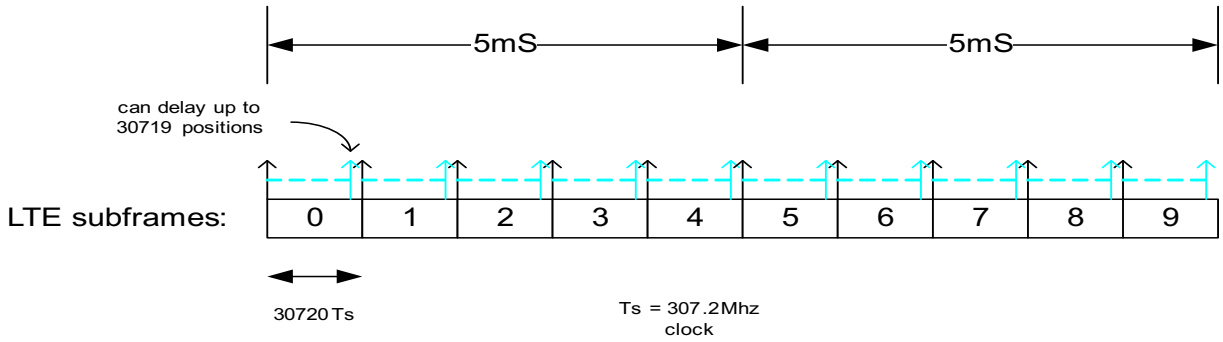


Figure 7-46 shows the LTE RAD timer count setup. It also only uses one clock count TC like WCDMA case.

LTE subframes (symbols) are 307200 clock (OBSAI) or 245760 clock (CPRI) periods in length. There are 10 subframes in a 10ms period. Each event could be used to get right timing to push 12 or 14 LTE symbols (LTE subframe size).

Events are generated from a slot boundary and may be delayed up to 307199 clocks.

Figure 7-47. LTE TDD Event (OBSAI)

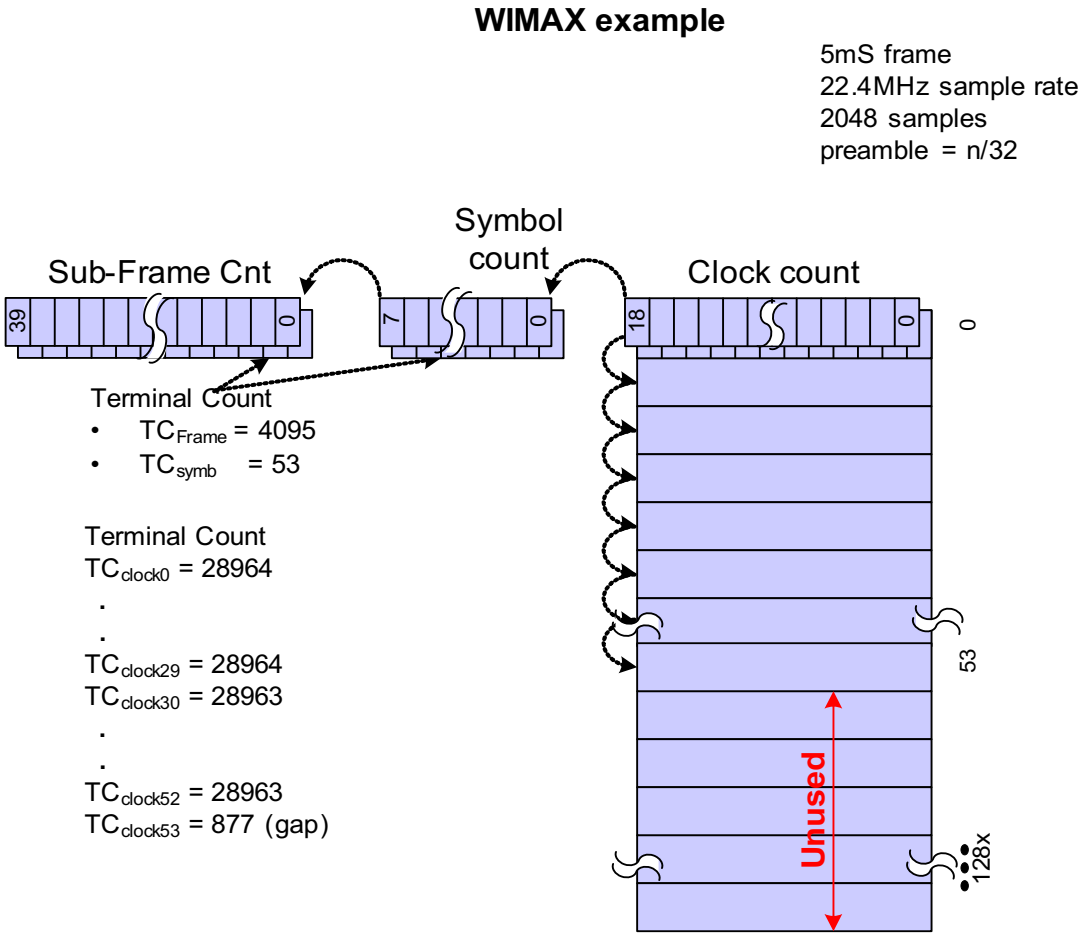


7.12.7 AT WiMAX Counter

The RADT timers have the flexibility needed for the various WIMAX rates. The following example is for a 5mS frame, 22.4MHz sample rate. The clock counter lut index will be programmed with the same value as the Symbol count TCsymb. Because the WIMAX sample clock is not an integer multiple of the 307.2MHz RADT timer clock, an approximation must be made for time-slot counts or for gap.

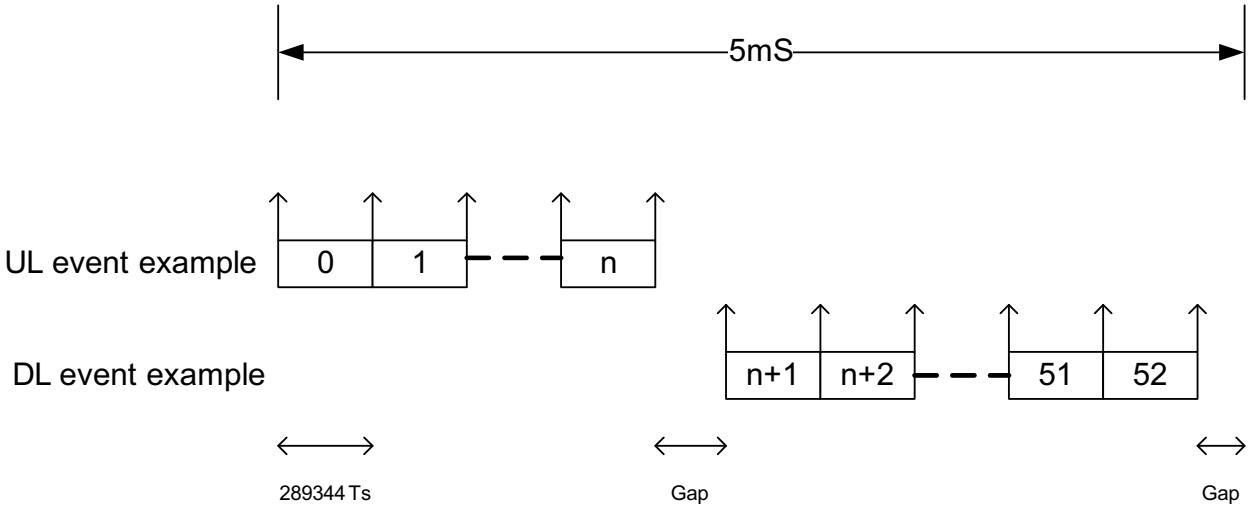
In the following example setup, error is accumulated in the gaps. Setup of counter clock lut may be modified to distribute the error among time-slots as is done in the GSM example.

Figure 7-48. WIMAX Timer Setup 2 ms Frame, 22.4 MHz Sample Rate



Events may be generated for uplink, downlink separately or both combined. Events would be triggered by symbol boundaries (OFDM packets). The event counter could be set to have one event per symbol. If separate events for uplink and downlink are desired, the mask for the particular event could be configured to select only uplink while masking out downlink events. The same could be done for downlink. When UL and DL ratio changes, the mask can be reprogrammed on the fly to change this event ratio.

Figure 7-49. WIMAX Events UL/DL Example

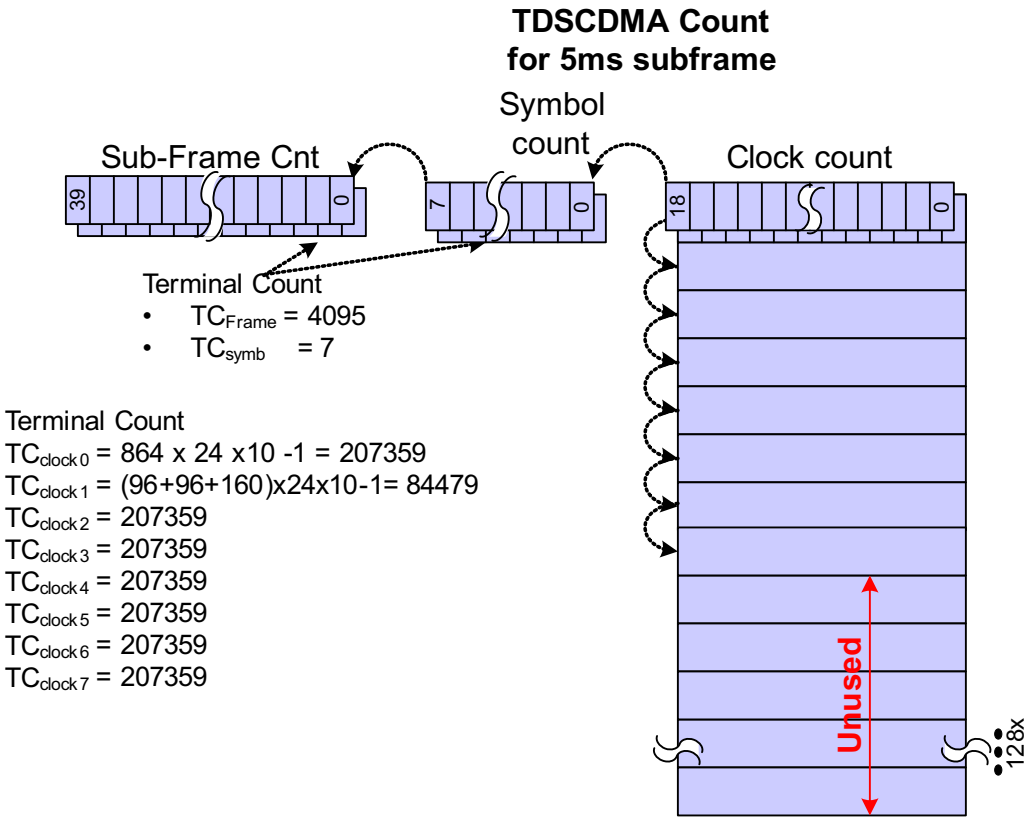


The example above shows two event counters generating UL and DL events separately

7.12.8 AT TD-SCDMA

The RADT timers will be programmed as shown for a 5mS TDSCDMA subframe. The clock counter lut index will be programmed with the same value as the Symbol count TCsymb.

Figure 7-50. RADT TDSCDMA Counter Programming



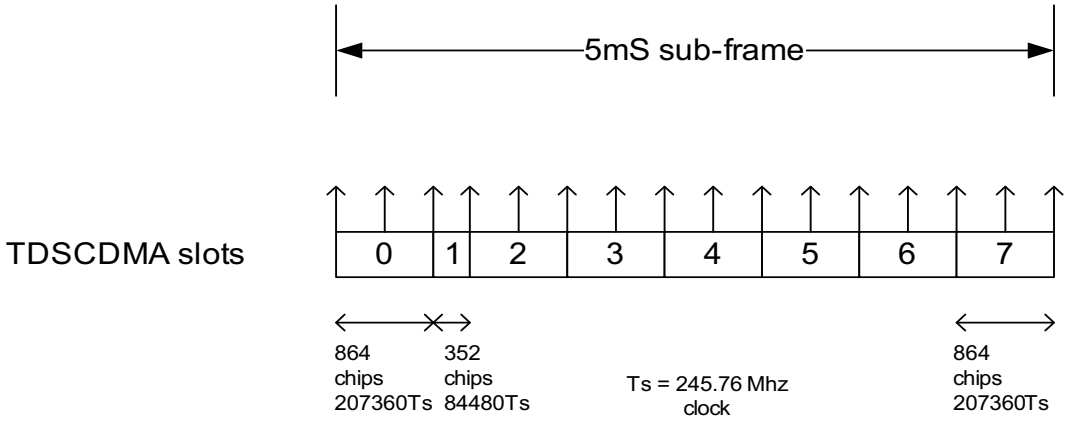
For TDSCDMA there are varying slot sizes. The lut is programmed to have the configuration shown. For example, the first terminal count for the clock counter is programmed to be 207359, the second location 84479, third location 207359, etc. This accounts for the varying but repeatable symbol sizes in TDSCDMA.

Symbol count TC is set to 7 in order to count eight symbols per subframe. Each subframe in this case is 5mS.

If a 10mS frame boundary is needed, the programmer has the option of doubling the depth used in the LUT and doubling the symbol count.

Events are generated on a ½ slot boundary as shown. Slot1 is a special slot, actually made up of DwPTS, GP and UpPTS. No event will be generated in the middle of this slot. The event counter will have an offset index or 0 and a period of 103680 clocks at 245.76 MHz byte clock. The slot boundary will be selected to trigger the event counter.

Figure 7-51. TDSCDMA Events (CPRI)

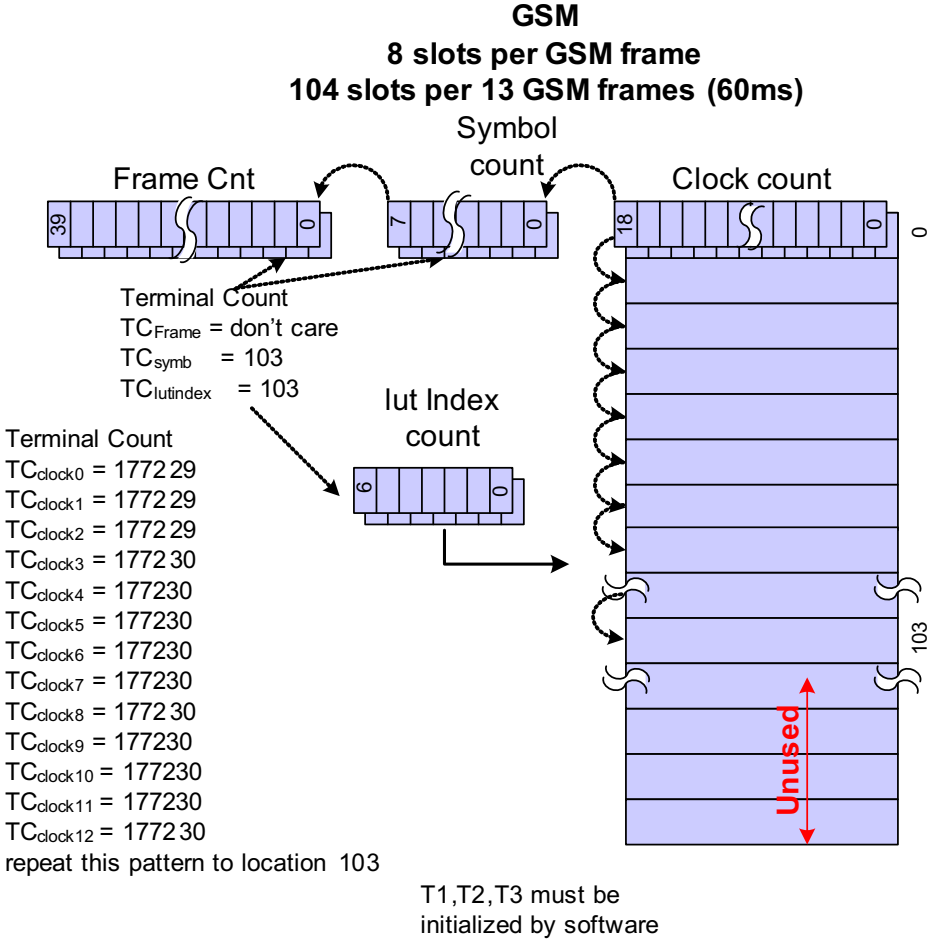


7.12.9 AT GSM Counter

Because the GSM sample clock is not an integer multiple of the 307.2MHz RADT timer clock, an approximation must be made for time-slot counts. Fortunately, every 60mS, the GSM timer is an exact multiple of the 307.2MHz clock. This allows for minor adjustments of the slot size, ± one clock. Because there are eight slots per GSM frame and there are 13 frames in 60 ms, there must be 104 timeslots (sample) Terminal Counts available in the clock count lookup table. The 128 locations of the symbol clock count lut can accommodate this. 104 locations are loaded with the varying slot sizes of either 177230 or 177231 clocks. In a series of 13 slots, three slots would count 177230 clocks and 10 slots would count 177231 clocks. This pattern of terminal counts will exactly allow for a 60mS count for 13 GSM frames.

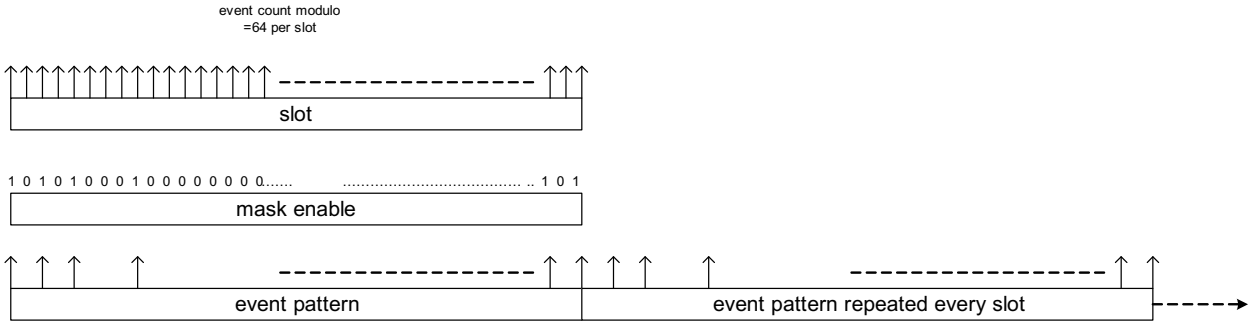
A unique aspect of setting up the RADT timers is that the symbol timer terminal count will be different from the lut index terminal count. Typically, these two terminal counts would be programmed to be equal. As mentioned in the previous paragraph the lut index needs to be able to increment and point to 104 locations, whereas the symbol counter needs to count eight slots per GSM frame. The 104 locations will account for clock count distribution among 13 GSM frames. T1, T2 and T3 counters will be initialized by software and will be used to count GSM frames. Typically the value in the Frame Count Field will not be used, rather the T1, T2, T3 values will be read by software. The wrap of the symbol count will indicate a GSM frame boundary.

Figure 7-52. GSM RADT Setup



Events will be generated based on slot boundaries where there will be 64 possible events per slot. The desired event counter will be programmed to be modulo64 with an index of 0. The 64-event mask will be programmed to enable the desired event pattern within the slot period. All slots for that particular event counter will have an identical event pattern.

Figure 7-53. GSM Mask Pattern

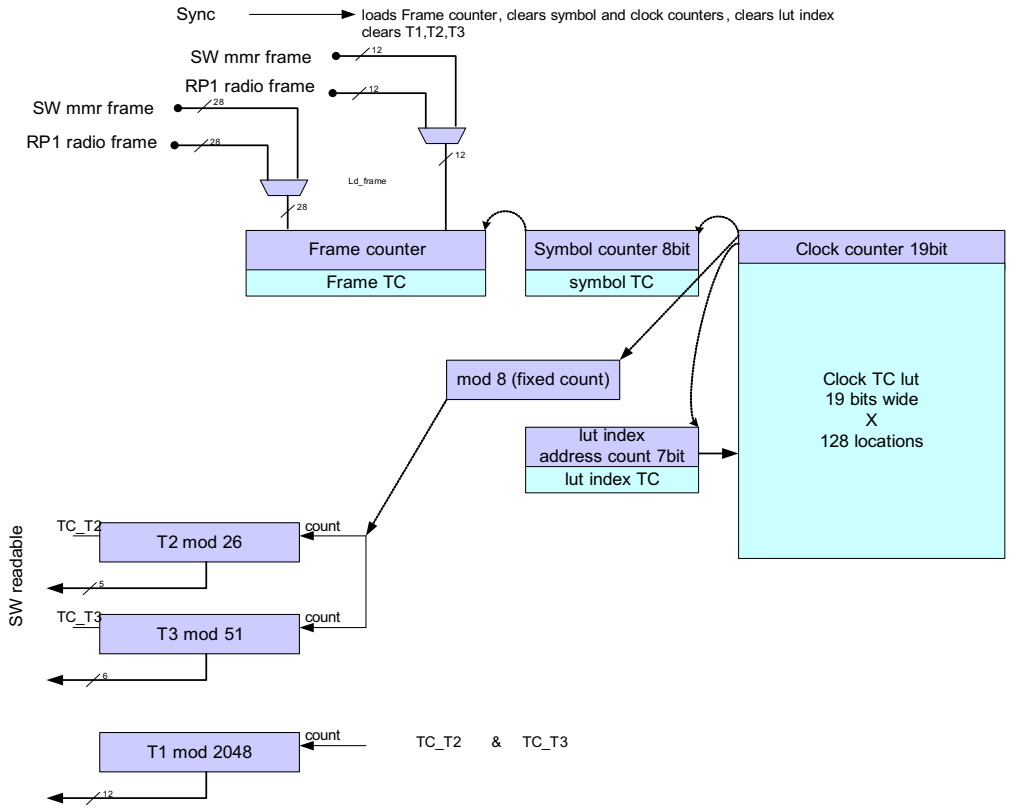


GSM also has another group of three timers that are incremented on a GSM frame by frame basis. They will be incremented every time the symbol (GSM slot) counter wraps. These are the T1, T2 and T3 timers fixed as:

- T1 = modulo 2048
- T2 = modulo 26
- T3 = modulo 51

These timers do not generate events but readable by software. Software must initializes the T1, T2, and T3 counters based on the current FCB received. Figure 7-54 shows how the T1,T2, and T3 counters work with the Radio timer.

Figure 7-54. GSM T counters with Radio timer



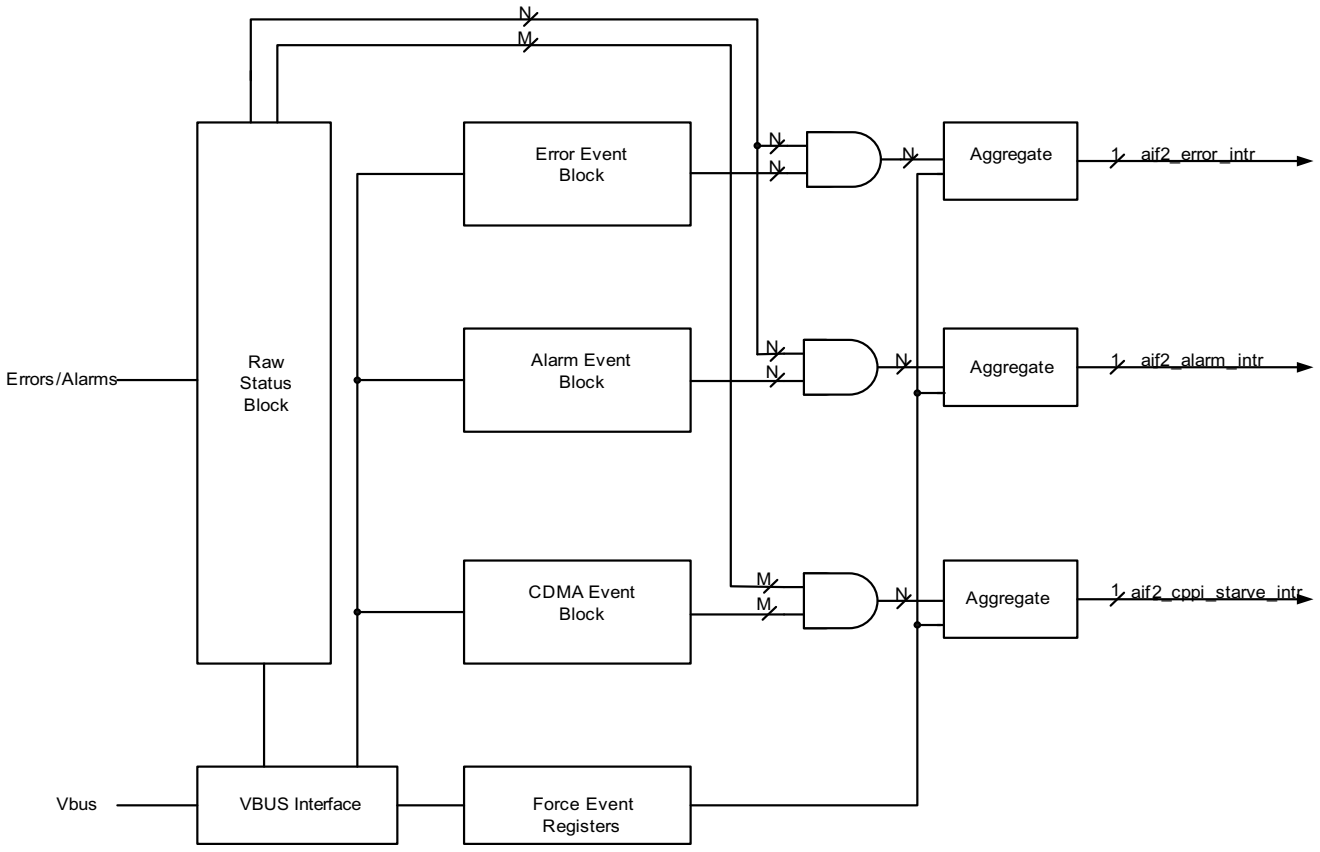
7.13 EE (Error and Exception Handler)

Terminology:

- **Error/alarm condition** signal is defined as the input to the EE module from its source module.
- **Interrupt** is defined as the output of the EE to the Interrupt Distributor.

The AIF2 has a large number of system error/alarm condition signals.The function of the Exception Interrupt Handler (EE) is to aggregate this large number of error/alarm condition signals to a smaller number of interrupts that can be used to generate DSP interrupts. Figure 7-55 shows the external interrupt generation functions of the EE.

Figure 7-55. EE Functional Block Diagram



7.13.1 EE Interrupt Support

The EE aggregates all of the error/alarm condition signals into three external system interrupts (**aif2_error_intr**, **aif2_alarm_intr**, **aif2_cppi_starve_intr**) that are broadcast as level high active signals to the interrupt distributor. These three system interrupts can be used as interrupts to the DSPs.

Each error/alarm condition signal is reported to the EE as a one *vbus_clk* or *sys_clock* wide pulse or a level (high active) from its source module and results in a unique bit being set to 1 in an Interrupt Raw Status Register (**IRS**) in the EE. This bit remains set until cleared by the DSP via the VBUSP configuration bus. All error/alarm condition signals, except the PKTDMA error/alarm condition signals, can be routed to either of the **aif2_error_intr** or **aif2_alarm_intr** interrupts, and the user can configure the subset of error/alarm condition signals that are associated with each of the **aif2_error_intr** and **aif2_alarm_intr** interrupts via individual enable registers (**EN_SET_EV**) for each of the signals inside the EE.

If the error/alarm condition signal is enabled via the enable registers (**EN_EV**), it will cause an interrupt on the **aif2_error_intr** or **aif2_alarm_intr** signal it is routed to when the error/alarm condition signal occurs. After the interrupt occurs on **aif2_error_intr** or **aif2_alarm_intr**, subsequent error/alarm condition signals that are received for this same **aif2_error_intr** or **aif2_alarm_intr** interrupt will not cause another interrupt until the DSP has acknowledged the interrupt by clearing the associated interrupt raw status register (**IRS_CLR**) and writing to the End of Interrupt Register (**EOI**).

All PKTDMA error/alarm condition signals can be routed to the **aif2_cppi_starve_intr** interrupt. The user can configure the subset of PKTDMA error/alarm condition signals that are associated with the **aif2_cppi_starve_intr** interrupt via individual enable registers (**EN_SET_EV**) for the PKTDMA signal inside the EE. If the PKTDMA error/alarm condition signal is enabled via the enable registers (**EN_EV**), it will cause an interrupt on the **aif2_cppi_starve_intr** signal when the error/alarm condition signal occurs. After the interrupt is generated, subsequent PKTDMA error/alarm condition signals that are received for this same **aif2_cppi_starve_intr** interrupt will not cause another interrupt until the DSP has acknowledged the interrupt clearing the associated interrupt raw status register (**IRS_CLR**) and by writing to the End of Interrupt Register (**EOI**).

Here are some additional rules that the EE follows:

- If an error/alarm condition signal is received and the corresponding error/alarm condition interrupt raw status bit is already set to 1, the occurrence of the second signal will not be saved.
- If the DSP attempts to clear an error/alarm condition at the same time that a new error/alarm condition signal is received on the same error/alarm condition input, the new error/alarm condition will cause the status bit to remain set to 1.

Table 7-27 provides a general description of the memory mapped registers (MMRs) that are provided to accomplish the EE functionality described above. Many copies of these registers are required to support the large number of error/alarm condition signals in the AIF2

Table 7-27. MMR General Description

	Description
Interrupt Raw Status (IRS)	This read only register contains the interrupt status bits for each error/status condition before any enable processing. **This Register not only shows error status but shows the operation status of the specific module as useful information for debugging**. EE Register map of IRS shows whether each bit field shows error or information. The interrupt status bits are set to 1 by an error/alarm condition or writing a 1 to the associated bit in the Interrupt Set (**IRS_SET**) register. They are cleared when a 1 is written to the associated bit in the Interrupt Clear (**IRS_CLR**) register
Interrupt Set (IRS_SET)	Write-only register. Writing a 1 to a bit will cause the associated bit in the Interrupt Raw Status (**IRS**) register to be set to 1. Used for debug to allow software to set an interrupt status bit. Writing a 0 has no effect.
Interrupt Clear (IRS_CLR)	Write-only register. Writing a 1 will cause the associated interrupt status bit in the Interrupt Raw Status (IRS) register to be cleared to a 0. Used by software to clear an interrupt condition after it has been serviced. Writing a 0 has no effect.
Interrupt Enabled Status (EN_STS_EV)	Read-only register that is the logical AND of the Interrupt Raw Status (IRS) register and the Interrupt Enable **(EN_EV)** register. This is only an address and not a physical register.
Interrupt Enable (EN_EV)	Read-only register that contains the enables for each of the interrupt status bits in the Interrupt Raw Status (**IRS**) register. An interrupt status bit can only cause an interrupt if it is enabled.
Interrupt Enable Set (EN_SET_EV)	Write-only register. Writing a 1 to a bit will cause the associated interrupt enable bit in the Interrupt Enable **(EN_EV)** register to be set to 1. Writing a 0 has no effect.
Interrupt Enable Clear (EN_CLR_EV)	Write-only register. Writing a 1 will cause the associated interrupt enable bit in the Interrupt Enable **(EN_EV)** register to be cleared to a 0 and disables the associated interrupt status bit. Writing a 0 has no effect.
End of Interrupt (EOI)	This register supports the End of Interrupt (**EOI**) interface between the AIF2 and the external Interrupt Distributor used in the Highlander 0,1 Interrupt Architecture. It is a common register that is used for servicing all of the AIF2 interrupts. This register is written by software to acknowledge the interrupt has been cleared so another interrupt of the same type can be generated. This register is written after any of the AIF2 interrupts has been cleared via the associated Interrupt Clear register.
Error/alarm condition Origination (ERR_ALRM_ORGN)	This register is used to help investigate which error/alarm condition source signal caused an interrupt.
AIF2 Run (EE_AIF2_RUN)	This register is used to give software an indication of the state AIF2 is in.
Interrupt Set (EE_VB_INTR_SET)	This register is use to force an interrupt on the output of the EE.
Interrupt Clear (EE_VB_INTR_CLR)	This register is use to clear a forced interrupt on the output of the EE.

7.13.2 Mapping Error/Alarm Conditions to Interrupts

7.13.2.1 Non-PKTDMA Error/Alarm Condition Mapping

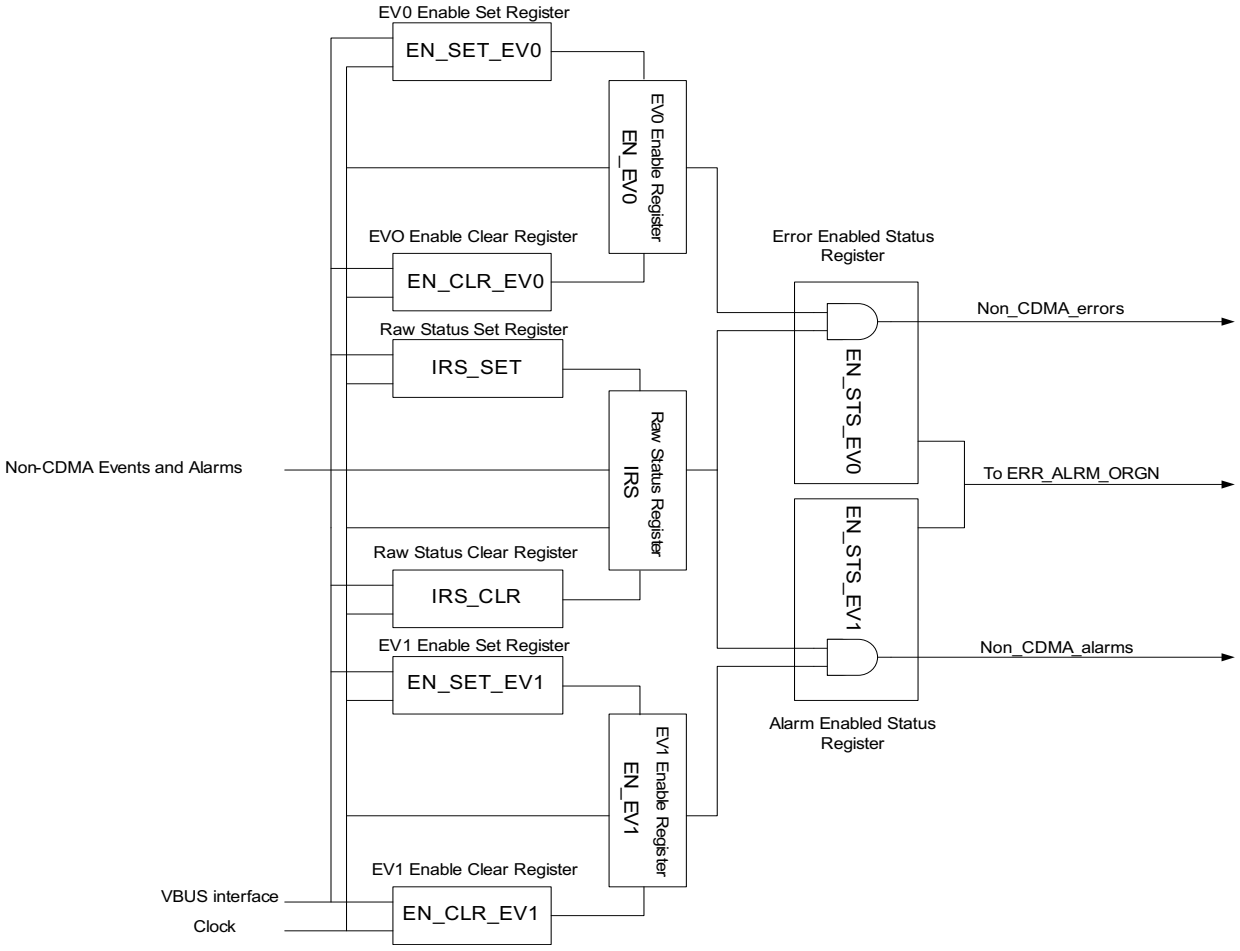
To map a non-PKTDMA error/alarm condition to the **aif2_error_intr** output interrupt the user would write a 1 to the enable bit corresponding to the error/alarm condition in the **EN_SET_EV0** register. This action would allow the error/alarm condition to pass to the **aif2_error_intr** aggregation logic. Mapping a non-PKTDMA error/alarm condition to **aif2_alarm_intr** output interrupt is done in the same manner described above except a 1 would be written to the **EN_SET_EV1** register.

If a user wanted to test the mapping of an error/alarm condition to the **aif2_error_intr** interrupt without actually generating the error/alarm condition signal from the source module, the user would write a 1 to the corresponding bit in the **EN_SET_EV0** register (to enable the bit to pass) and then write a 1 to the bit in the **IRS_SET** register. The **IRS** bit corresponding to the error/alarm condition would then have to be cleared from the **IRS** by writing a 1 to the corresponding bit in the **IRS_CLR** register. Finally the user would write the **EOI** register to acknowledge the servicing of the interrupt.

The same process can be applied to map an error/alarm condition to the **aif2_alarm_intr** interrupt by writing a 1 to the corresponding bit in the **EN_SET_EV1** register. The **IRS** bit corresponding to the error/alarm condition would then have to be cleared from the **IRS** by writing a 1 to the corresponding bit in the **IRS_CLR** register. Finally the user would write the **EOI** register to acknowledge the servicing of the interrupt.

Figure 7-56 shows a block diagram showing the mapping logic of non-PKTDMA error/alarm conditions.

Figure 7-56. Non-PKTDMA Error/Alarm Conditions



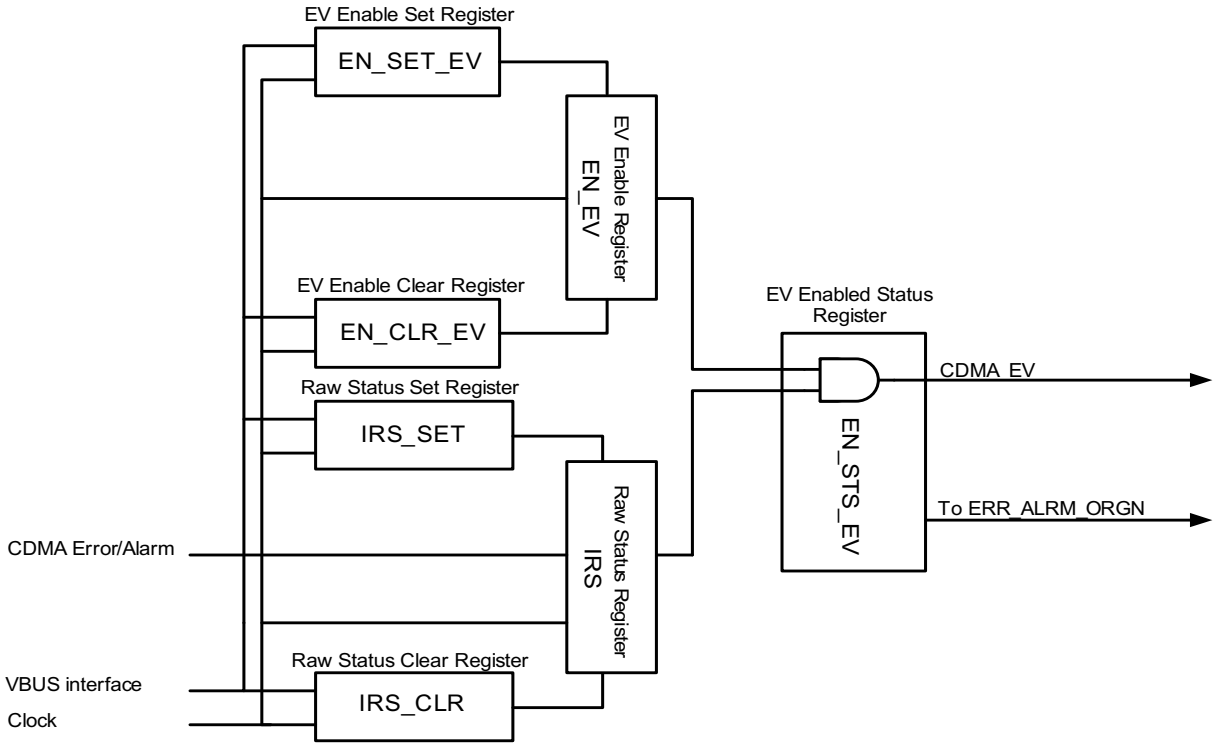
7.13.2.2 PKTDMA Error/Alarm Condition Mapping

To map a PKTDMA error/alarm condition to the **aif2_cppi_starve_intr** interrupt the user would write a 1 to the enable bit corresponding to the error/alarm condition in the **EN_SET_EV** register. This action would allow the error/alarm condition to pass to the **aif2_cppi_starve_intr** aggregation logic. (PKTDMA error/alarm conditions cannot be mapped to **aif2_error_intr** or **aif2_alarm_intr** interrupts.)

If a user wanted to test the mapping of an error/alarm condition to the **aif2_cppi_starve_intr** interrupt without actually generating the error/alarm condition signal from the source module, the user would write a 1 to the corresponding bit in the **EN_SET_EV** register (to enable the bit to pass) and then write a 1 to the bit in the **IRS_SET** register. The **IRS** bit corresponding to the error/alarm condition would then have to be cleared from the **IRS** by writing a 1 to the corresponding bit in the **IRS_CLR** register. Finally the user would write the **EOI** register to acknowledge the servicing of the interrupt.

Figure 7-57 shows a block diagram showing the mapping of PKTDMA error/alarm conditions.

Figure 7-57. PKTDMA Error/Alarm Conditions



7.13.3 End-of-Interrupt Support (EOI)

A transaction is initiated on the interface each time **AIF2_EOI** is written, but it is delayed for at least one vbus clock cycle after the write completes. An additional delay is incurred if there is a previous clear of any of the Raw Status (**IRS_CLR**) registers that have not yet taken effect at the AIF2 interrupt outputs. In this case, it is important to wait to prevent a false interrupt from being generated by the Interrupt Distributor Component when the EOI transaction is initiated.

The value of the **eoi_vector** when the **eoi_write** signal pulses high on the EOI interface identifies the interrupt that is being acknowledged. The **eoi_vector** value is 8 bits wide and follows the value in the *EOI_VECTOR* field of **EOI**. The *EOI_VECTOR* values written by the DSP to identify the interrupt that is being acknowledged is defined by the system. Normally, this EOI_VECTOR value for AIF2 is zero.Writes to **EOI** have no effect on the internal interrupt processing of the AIF2.

7.13.4 EE Run (EE_AIF2_RUN)

The EE contains an **EE_AI2_RUN** register that has two bits named **aif2_phy_run** and **aif2_global_run.** When the **aif2_phy_run** and **aif2_global_run** bits are set to 1, it indicates to the DSP software that AIF2 is operating normally. If **aif2_phy_run** or *aif2_global_run* are cleared to a 0, it indicates that AIF2 is not running normally.

When the UL, DL, or application software is rebooted for a given device, the software does not know the state of the current system. The software polls each peripheral to detect the current system state and to determine whether the peripherals need to be initialized or the software needs to realign itself to peripherals that are running correctly. For AIF2, the rebooted software will not want to crash the whole daisy chain and will use **aif2_phy_run** and *aif2_global_run* bits to detect the state of the AIF2.

The **aif2_phy_run** bit is controlled as follows:

- Cleared when the AIF2 is reset
- Cleared whenever an enabled error/alarm condition in the physical clock domain causes an interrupt.
- Set by DSP software (Note that if there is an active interrupt caused by an error/alarm condition in the physical clock domain at the same time the DSP sets **aif2_phy_run**, the bit will be cleared instead.)

The **aif2_global_run** bit is controlled as follows:

- Cleared when the AIF2 is reset
- Cleared whenever any enabled error/alarm condition, in either the VBUS clock domain or the SYS clock domain, causes an interrupt.
- Set by DSP software (Note that if there is an active interrupt caused by an error/alarm condition in either clock domain at the same time the DSP sets **aif2_global_run**, the bit will be cleared instead.)

7.13.5 Error/Alarm Condition Force Support

The EE module contains registers, **EE_VB_INTR_SET** and **EE_VB_INTR_CLR**, that can be used to force an interrupt on any of the interrupt output signals. Each register contains three bits. One bit of each to the three interrupt output signal. If the user would like to force an interrupt to occur, the user would write a 1 to the bit in the **EE_VB_INTR_SET** register corresponding to the interrupt that is to be set. The interrupt will remain set until the user clears that interrupt by writing a 1 to the corresponding bit in the **EE_VB_INTR_CLR** register.

7.13.6 Error/Alarm Condition Origination Support

The EE module contains a register, **ERR_ALRM_ORGN**, which is designed to help the user minimize the number of register reads required to investigate which error/alarm condition caused an interrupt to occur. This register contains an indication of which **EN_STS_EV** register contains an error/alarm condition that may have caused the initial interrupt.

Table 7-28 shows which bits in the **ERR_ALRM_ORGN** register correspond to which **EN_STS_EV** register.

Table 7-28. ERR_ALARM_ORGN Bit Definitions

Register Name	Bit
ee_lk_en_sts_a[0]	0
ee_lk_en_sts_b[0]	1
ee_lk_en_sts_a[1]	2
ee_lk_en_sts_b[1]	3
ee_lk_en_sts_a[2]	4
ee_lk_en_sts_b[2]	5
ee_lk_en_sts_a[3]	6
ee_lk_en_sts_b[3]	7
ee_lk_en_sts_a[4]	8
ee_lk_en_sts_b[4]	9

Table 7-28. ERR_ALARM_ORGN Bit Definitions (continued)

Register Name	Bit
ee_lk_en_sts_a[5]	10
ee_lk_en_sts_b[5]	11
ee_at_en_sts	12
ee_sd_en_sts	13
ee_db_en_sts	14
ee_ad_en_sts	15
ee_cd_en_sts	16
ee_vc_en_sts	17

The **ERR_ALRM_ORGN** register is a read only register that is updated every clock cycle. If an error/alarm condition is received in a raw status register (**IRS**) and enabled via the enabled status register (**EN_STS_EV**) after another error/alarm condition has caused an interrupt and that original error/alarm condition has not been cleared there could be two (or more) bits set in the **ERR_ALRM_ORGN** register.

Without the **ERR_ALRM_ORGN** register the user would have to read every **EN_STS_EV** register to determine which error/alarm condition caused an interrupt. In this device, it would be 17 reads. With the **ERR_ALRM_ORGN** register the user can accomplish the same determination with two reads. The user would first read the **ERR_ALRM_ORGN** register and see which bit is set. Using Table 4, the user would then know which **EN_STS_EV** register contained the error/alarm condition that caused the interrupt. The user would then read that **EN_STS_EV** register to determine which error/alarm condition was set.

Example:

The **at_ee_rp1_rp3_frm_err** error/alarm condition signal is mapped so that if the error/alarm condition were to occur it would cause and interrupt on **aif2_error_intr.**

The **at_ee_rp1_rp3_frm_err** signal toggles and is latched into the **EE_AT_IRS** register setting bit 1 in that register. Because it was enabled in the **EE_AT_EN_STS_EV0** register, the signal is passed to the aggregation logic and the interrupt happens on the **aif2_error_intr** output signal.

Seeing the interrupt, the user would now read the **ERR_ALRM_ORGN** register and find that bit 12 is set. From Table 7-28, this bit corresponds to the **EE_AT_EN_STS** register.

The user would then read the **EE_AT_EN_STS** register and see that bit 1 was set. From the register description in the EE Register Memory Map section of this document the user would see that bit 1 corresponds to **at_ee_rp1_rp3_frm_err** input.

Register Map

The base address for AIF2 is **0x01F00000**. The PKTDMA module register map is in the *Multicore Navigator for Keystone Devices User Guide* (SPRUGR9). If there's no special comment in the description, the default value should be zero.

Topic **Page**

8.1 VC Registers 217
8.2 SD Registers 222
8.3 RM Registers 241
8.4 TM Registers 254
8.5 CI Registers 274
8.6 CO Registers 275
8.7 RT Registers 276
8.8 PD Registers 281
8.9 PE Registers 314
8.10 DB Registers 349
8.11 AD Registers 385
8.12 AT Registers 421
8.13 EE Registers 503

8.1 VC Registers

Table 8-1. VC Register Memory Map

Offset	Register Name	Description	Section
0x0000	aif2_pid	aif2 Peripheral ID Register	Section 8.1.1.1
0x0004	aif2_scratch	aif2 VC Scratch Register	Section 8.1.1.2
0x0008	aif2_reset	aif2 Reset Register	Section 8.1.1.3
0x000C	aif2_emu	aif2 Emulation Control Register	Section 8.1.1.4
0x0010	vc_stat	VC Status Register	Section 8.1.1.5

8.1.1 Grouped Common Registers Details

8.1.1.1 aif2_pid Register (Address = 0x0000)

Table 8-2. AIF2 Peripheral ID Register Field Descriptions

Bits	Field Name	Type	Description
31-30	scheme	READ	Current scheme
29-28	rsvd	NOT_ACCESSIBLE	RESERVED
27-16	func	READ	Function code assigned to AIF2
15-11	RTL	READ	RTL Version R code
10-8	major	READ	Major revision X code
7-6	custom	READ	Custom version code
5-0	minor	READ	Minor revision Y code

8.1.1.2 aif2_scratch Register (Address = 0x0004)

Table 8-3. AIF2 Scratch Register Field Descriptions

Bits	Field Name	Type	Description
31-0	scratch	READ_WRITE	Scratch Field for software debugging purpose

8.1.1.3 aif2_reset Register (Address = 0x0008)

Table 8-4. AIF2 Software Reset Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	sw_rst	WRITE	AIF2 software reset pulse 1 = Set reset pulse

8.1.1.4 aif2_emu Register (Address = 0x000C)

Table 8-5. AIF2 Emulation Control Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	RESERVED
2	rt_sel	READ_WRITE	RT_SEL bit 0 = AIF2 emulation mode is controlled only by the aif2_emu_dbgsusp CBA signal. 1 = AIF2 emulation mode is controlled only by the aif2_emu_dbgsusp_rt CBA signal.
1	soft	READ_WRITE	SOFT bit 0 = Disable auto trigger modeAIF2 halts as soon as it can in AIF2 Interface State Machine states 0, 1, 5, or 8. 1 = AIF2 halts gracefully and completes any decodes in progress. Halts in AIF2 Interface State Machine states 0, 1, or 5
0	freerun	READ_WRITE	FREERUN bit 0 = AIF2 responds to the emulation suspend signal it is monitoring and operates according to setting of the SOFT bit. 1 = AIF2 ignores emulation suspend signals and runs to completion

8.1.1.5 vc_stat Register (Address = 0x0010)

Table 8-6. VC Status Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	emu_halt	READ	Emulation halt status

8.2 SD Registers

Table 8-7. SD Register Memory Map

Offset	Register Name	Access	Description	Section
0x8000	SD_RX_EN_CFG[0]	R/W	RX Enable Register (Link 0)	Section 8.2.1.1
0x8004	SD_RX_R1_CFG[0]	R/W	RX Configuration Register	Section 8.2.1.2
0x8008	SD_RX_R2_CFG[0]	R/W	RX Configuration Register	Section 8.2.1.3
0x800C	SD_RX_STS[0]	R	RX Status Register	Section 8.2.1.4
0x8010	SD_TX_EN_CFG[0]	R/W	TX Enable Register	Section 8.2.1.5
0x8014	SD_TX_R1_CFG[0]	R/W	TX Configuration Register	Section 8.2.1.6
0x8018	SD_TX_R2_CFG[0]	R/W	TX Configuration Register	Section 8.2.1.7
0x801C	SD_TX_STS[0]	R/W	TX Status Register	Section 8.2.1.8
0x8800	SD_RX_EN_CFG[1]	R/W	RX Enable Register (Link 1)	Section 8.2.1.1
0x8804	SD_RX_R1_CFG[1]	R/W	RX Configuration Register	Section 8.2.1.2
0x8808	SD_RX_R2_CFG[1]	R/W	RX Configuration Register	Section 8.2.1.3
0x880C	SD_RX_STS[1]	R	RX Status Register	Section 8.2.1.4
0x8810	SD_TX_EN_CFG[1]	R/W	TX Enable Register	Section 8.2.1.5
0x8814	SD_TX_R1_CFG[1]	R/W	TX Configuration Register	Section 8.2.1.6
0x8818	SD_TX_R2_CFG[1]	R/W	TX Configuration Register	Section 8.2.1.7
0x881C	SD_TX_STS[1]	R/W	TX Status Register	Section 8.2.1.8
0x9000	SD_RX_EN_CFG[2]	R/W	RX Enable Register (Link 2)	Section 8.2.1.1
0x9004	SD_RX_R1_CFG[2]	R/W	RX Configuration Register	Section 8.2.1.2
0x9008	SD_RX_R2_CFG[2]	R/W	RX Configuration Register	Section 8.2.1.3
0x900C	SD_RX_STS[2]	R	RX Status Register	Section 8.2.1.4
0x9010	SD_TX_EN_CFG[2]	R/W	TX Enable Register	Section 8.2.1.5
0x9014	SD_TX_R1_CFG[2]	R/W	TX Configuration Register	Section 8.2.1.6
0x9018	SD_TX_R2_CFG[2]	R/W	TX Configuration Register	Section 8.2.1.7
0x901C	SD_TX_STS[2]	R/W	TX Status Register	Section 8.2.1.8
0x9800	SD_RX_EN_CFG[3]	R/W	RX Enable Register (Link 3)	Section 8.2.1.1
0x9804	SD_RX_R1_CFG[3]	R/W	RX Configuration Register	Section 8.2.1.2
0x9808	SD_RX_R2_CFG[3]	R/W	RX Configuration Register	Section 8.2.1.3
0x980C	SD_RX_STS[3]	R	RX Status Register	Section 8.2.1.4
0x9810	SD_TX_EN_CFG[3]	R/W	TX Enable Register	Section 8.2.1.5
0x9814	SD_TX_R1_CFG[3]	R/W	TX Configuration Register	Section 8.2.1.6
0x9818	SD_TX_R2_CFG[3]	R/W	TX Configuration Register	Section 8.2.1.7
0x981C	SD_TX_STS[3]	R/W	TX Status Register	Section 8.2.1.8
0xA000	SD_RX_EN_CFG[4]	R/W	RX Enable Register (Link 4)	Section 8.2.1.1
0xA004	SD_RX_R1_CFG[4]	R/W	RX Configuration Register	Section 8.2.1.2
0xA008	SD_RX_R2_CFG[4]	R/W	RX Configuration Register	Section 8.2.1.3
0xA00C	SD_RX_STS[4]	R	RX Status Register	Section 8.2.1.4
0xA010	SD_TX_EN_CFG[4]	R/W	TX Enable Register	Section 8.2.1.5
0xA014	SD_TX_R1_CFG[4]	R/W	TX Configuration Register	Section 8.2.1.6
0xA018	SD_TX_R2_CFG[4]	R/W	TX Configuration Register	Section 8.2.1.7
0xA01C	SD_TX_STS[4]	R/W	TX Status Register	Section 8.2.1.8
0xA800	SD_RX_EN_CFG[5]	R/W	RX Enable Register (Link 5)	Section 8.2.1.1
0xA804	SD_RX_R1_CFG[5]	R/W	RX Configuration Register	Section 8.2.1.2
0xA808	SD_RX_R2_CFG[5]	R/W	RX Configuration Register	Section 8.2.1.3
0xA80C	SD_RX_STS[5]	R	RX Status Register	Section 8.2.1.4
0xA810	SD_TX_EN_CFG[5]	R/W	TX Enable Register	Section 8.2.1.5

Table 8-7. SD Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0xA814	SD_TX_R1_CFG[5]	R/W	TX Configuration Register	Section 8.2.1.6
0xA818	SD_TX_R2_CFG[5]	R/W	TX Configuration Register	Section 8.2.1.7
0xA81C	SD_TX_STS[5]	R/W	TX Status Register	Section 8.2.1.8
0xB000	SD_PLL_B8_EN_CFG		B8 PLL configuration Register	Section 8.2.1.9
0xB004	SD_PLL_B4_EN_CFG		B4 PLL configuration Register	Section 8.2.1.10
0xB008	SD_PLL_B8_CFG		B8 PLL configuration Register	Section 8.2.1.11
0xB00C	SD_PLL_B4_CFG		B4 PLL configuration Register	Section 8.2.1.12
0xB010	SD_PLL_B8_STS		B8 PLL Status Register	Section 8.2.1.13
0xB014	SD_PLL_B4_STS		B4 PLL Status Register	Section 8.2.1.14
0xB018	SD_CLK_SEL_CFG		Clock selection	Section 8.2.1.15
0xB01C	SD_LK_CLK_DIS_CFG		Link clock Enable Register	Section 8.2.1.16

8.2.1 Grouped Common Registers Details

8.2.1.1 SD_RX_EN_CFG[0] Register (offset = 0x8000)

Table 8-8. SD_RX_EN_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	ENRX	READ_WRITE	The receiver SerDes macro should be enabled after the SerDes configurations are complete and the PLL has acquired lock. 0 = Disable RX 1 = Enable RX
SD_RX_EN_CFG[1]	**Access = READ_WRITE**		**Address [0x8800]**
SD_RX_EN_CFG[2]	**Access = READ_WRITE**		**Address [0x9000]**
SD_RX_EN_CFG[3]	**Access = READ_WRITE**		**Address [0x9800]**
SD_RX_EN_CFG[4]	**Access = READ_WRITE**		**Address [0xA000]**
SD_RX_EN_CFG[5]	**Access = READ_WRITE**		**Address [0xA800]**

8.2.1.2 SD_RX_R1_CFG[0] Register (offset = 0x8004)

Table 8-9. SD_RX_R1_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-15	RSVD3	NOT_ACCESSIBLE	Reserved.
14-12	CDR	READ_WRITE	Clock/data recovery. Configures the clock/data recovery algorithm. • 000 Second order, threshold 15 • 001 Second order, threshold 7 • 010 Second order, threshold 3 • 011 Second order, threshold 1 (Suitable for asynchronous system where crystals are different) • 100 First order, threshold 15 • 101 First order, threshold 3 • 110 First order, threshold 1(Suitable for synchronous system where the same clock source is used) • 111 First order, threshold 7 (Suitable for short reach synchronous chip to chip applications)
11	RSVD2	NOT_ACCESSIBLE	Reserved.
10-8	LOS	READ_WRITE	The LOSi bits should be set to 2b10 to allow Loss of signal detection by the SerDes macro. For SD internal loopback, this should be disabled. • 000 Disabled • 100 Enabled
7-6	RSVD1	NOT_ACCESSIBLE	Reserved.
5-4	ALIGN	READ_WRITE	The receiver frame synchronizer must control the byte alignment feature of each receiver so that a jog can be initiated based on detection of non-alignment. The ALIGN bits must be set to 2b00 for normal operation. • 00 Alignment Disabled • 01 Comma alignment enabled • 10 Alignment Jog (The symbol alignment will be adjusted by one bit position when this mode is selected)
3-2	RSVD	NOT_ACCESSIBLE	Reserved.
1-0	RXRATE	READ_WRITE	The AIF2 antenna interface link configuration registers must reflect the sample rate of each receiver as 2x, 4x, 8x so that the receiver logic is aware of the rate setting. These rates are listed in the following table. • 01 8x • 10 4x or 5x for CPRI • 11 2x

SD_RX_R1_CFG[1]	**Access = READ_WRITE**	**Address [0x8804]**
SD_RX_R1_CFG[2]	**Access = READ_WRITE**	**Address [0x9004]**
SD_RX_R1_CFG[3]	**Access = READ_WRITE**	**Address [0x9804]**
SD_RX_R1_CFG[4]	**Access = READ_WRITE**	**Address [0xA004]**
SD_RX_R1_CFG[5]	**Access = READ_WRITE**	**Address [0xA804]**

8.2.1.3 SD_RX_R2_CFG[0] Register (offset = 0x8008)

Table 8-10. SD_RX_R2_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-25	RSVD5	NOT_ACCESSIBLE	Reserved.
24-22	RXTESTPATT	READ_WRITE	Enables and selects test patterns. Enables and selects verification of one of three PRBS patterns, a user defined pattern, or a clock test pattern. 000 Test mode disabled 001 Alternating 0/1 pattern 010 Generate or verify 2^7– 1 PRBS 011 Generate or verify 2^23– 1 PRBS 100 Generate or verify 2^31 – 1 PRBS
21-17	RSVD4	NOT_ACCESSIBLE	Reserved.
16-15	RXLOOPBACK	READ_WRITE	The receiver frame synchronizer must have knowledge that each SerDes port had been placed in Tx to Rx loopback so that the receiver can suppress events due to the realignment of the received frames. These bits should be set to 2b00 for normal operation. 00 Disabled 11 Loopback
14-12	RSVD3	NOT_ACCESSIBLE	Reserved.
11	ENOC	READ_WRITE	Enable Offset Compensation. Enables samplers offset compensation. 0 Disabled 1 Enabled
10	RSVD2	NOT_ACCESSIBLE	Reserved.
9	EQHLD	READ_WRITE	Hold Equalizer. Holds equalizer in the current state. 0 EQ adaptation enabled 1 EQ adaptation held
8-6	EQ	READ_WRITE	Equalizer. Enables and configures the equalizer to compensate for loss in the transmission media. 000 No Equalization 001 Fully adaptive equalization 010 Precursor equalization analysis 011 Post cursor equalization analysis
5	RSVD1	NOT_ACCESSIBLE	Reserved.
4-2	RXTERM	READ_WRITE	Termination. Selects the input termination options suitable for AC or DC coupled scenarios. 000 Common point connected to VDDT 001 Common point set to 0.7VDDT 011 Common point floating
1	RSVD	NOT_ACCESSIBLE	Reserved.
0	RXINVPAR	READ_WRITE	Invert Polarity. Inverts the polarity of RXPi and RXNi. 0 Normal polarity 1 Inverted polarity

SD_RX_R2_CFG[1]	**Access = READ_WRITE**	**Address [0x8808]**
SD_RX_R2_CFG[2]	**Access = READ_WRITE**	**Address [0x9008]**
SD_RX_R2_CFG[3]	**Access = READ_WRITE**	**Address [0x9808]**
SD_RX_R2_CFG[4]	**Access = READ_WRITE**	**Address [0xA008]**
SD_RX_R2_CFG[5]	**Access = READ_WRITE**	**Address [0xA808]**

8.2.1.4 SD_RX_STS[0] Register (offset = 0x800C)

Table 8-11. SD_RX_STS[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-25	RSVD4	NOT_ACCESSIBLE	Reserved.
24	RXTESTFAIL	READ	Test Failure. Driven high when an error is encountered during a test sequence executed on an individual channel. Synchronous to RXBCLK.
23-19	RSVD3	NOT_ACCESSIBLE	Reserved.
18-16	RXBUSWIDTH	READ	The receiver bus bandwidth is fixed in hardware to 20 bits and is not configurable through an MMR. However the value that reflects a 20-bit bus width will be read of the configuration register is read. 010 20-bit operation 011 16-bit operation
15-11	RSVD2	NOT_ACCESSIBLE	Reserved.
10	EQOVER	READ	Driven high during equalizer analysis if over equalized.
9	EQUNDER	READ	Driven high during equalizer analysis if under equalized.
8	OCIP	READ	Offset compensation in progress. Driven high asynchronously during offset compensation.
7-3	RSVD1	NOT_ACCESSIBLE	Reserved.
2	LOSDTCT	READ	The receiver frame synchronizer must have knowledge that each SerDes port had detected a loss of signal condition so that the receiver can suppress events due to a loss of frame synchronization. 0 disabled 1 enabled
1	RSVD	NOT_ACCESSIBLE	Reserved.
0	SYNC	READ	If the alignment feature of the SerDes is used by hardware, the receiver frame synchronizer must have knowledge that each SerDes port had completed a requested byte alignment so that the byte alignment control logic can operate. 0 Alignment disabled (after getting frame sync, the synchronizer is in idle state) 1 Alignment enabled (when synchronizer is trying to detect K characters)

SD_RX_STS[1]	**Access = READ_WRITE**	**Address [0x880C]**
SD_RX_STS[2]	**Access = READ_WRITE**	**Address [0x900C]**
SD_RX_STS[3]	**Access = READ_WRITE**	**Address [0x980C]**
SD_RX_STS[4]	**Access = READ_WRITE**	**Address [0xA00C]**
SD_RX_STS[5]	**Access = READ_WRITE**	**Address [0xA80C]**

8.2.1.5 SD_TX_EN_CFG[0] Register (offset = 0x8010)

Table 8-12. SD_TX_EN_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	ENTX	READ_WRITE	The receiver SerDes macro should be enabled once all the SerDes configurations are complete, and the PLL has acquired lock. 0 = Disable TX 1 = Enable TX

SD_TX_EN_CFG[1]	**Access = READ_WRITE**	**Address [0x8810]**
SD_TX_EN_CFG[2]	**Access = READ_WRITE**	**Address [0x9010]**
SD_TX_EN_CFG[3]	**Access = READ_WRITE**	**Address [0x9810]**
SD_TX_EN_CFG[4]	**Access = READ_WRITE**	**Address [0xA010]**
SD_TX_EN_CFG[5]	**Access = READ_WRITE**	**Address [0xA810]**

8.2.1.6 SD_TX_R1_CFG[0] Register (offset = 0x8014)

Table 8-13. SD_TX_R1_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-13	RSVD3	NOT_ACCESSIBLE	Reserved.
12-10	TXTESTPATT	READ_WRITE	Enables and selects test patterns. Enables and selects verification of one of three PRBS patterns, a user defined pattern, or a clock test pattern. 000 Test mode disabled 001 Alternating 0/1 pattern 010 Generate or verify 2^7– 1 PRBS 011 Generate or verify 2^23– 1 PRBS 100 Generate or verify 2^31– 1 PRBS
9-7	RSVD2	NOT_ACCESSIBLE	Reserved.
6-5	TXLOOPBACK	READ_WRITE	Loopback. Enable loopback high to assert. 00 Disabled 11 Loopback
4	RSVD1	NOT_ACCESSIBLE	Reserved.
3	MSYNC	READ_WRITE	Synchronization Master. Enables the channel as the master lane for synchronization purposes. Tie high for master lane (normally link 0) and tie low for slave lanes (other links) 0 Disable 1 Enable
2	RSVD	NOT_ACCESSIBLE	Reserved.
1-0	TXRATE	READ_WRITE	AIF2 utilizes the system clock of 307.2 MHz generated by one of the transmit links, all the links must be set to 8x to activate any internal module rate (2x, 4x, 5x, 8x) and at least this field should be set to 0x1 to activate other module before activating TX SerDes 01 8x (only 8x should be set for Tx SerDes)

SD_TX_R1_CFG[1]	**Access = READ_WRITE**	**Address [0x8814]**
SD_TX_R1_CFG[2]	**Access = READ_WRITE**	**Address [0x9014]**
SD_TX_R1_CFG[3]	**Access = READ_WRITE**	**Address [0x9814]**
SD_TX_R1_CFG[4]	**Access = READ_WRITE**	**Address [0xA014]**
SD_TX_R1_CFG[5]	**Access = READ_WRITE**	**Address [0xA814]**

8.2.1.7 SD_TX_R2_CFG[0] Register (offset = 0x8018)

Table 8-14. SD_TX_R2_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD4	NOT_ACCESSIBLE	Reserved.
17	FIRUPT	READ_WRITE	Transmitter pre and post cursor FIR filter Update. Update control of FIR taps weights. Fields TWPRE and TWPST1 can be updated when TXBCLK and this input are both high.
16	RSVD3	NOT_ACCESSIBLE	Reserved.
15-11	TWPST	READ_WRITE	Adjacent post cursor Tap Weight. Selects one of 32 output tap weights for TX waveform conditioning. Positive tap weight (%) Negative tap weight (%) 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 +2.5 1 0 0 0 1 -2.5 0 0 0 1 0 +5.0 1 0 0 1 0 -5.0 0 0 0 1 1 +7.5 1 0 0 1 1 -7.5 0 0 1 0 0 +10.0 1 0 1 0 0 -10.0 0 0 1 0 1 +12.5 1 0 1 0 1 -12.5 0 0 1 1 0 +15.0 1 0 1 1 0 -15.0 0 0 1 1 1 +17.5 1 0 1 1 1 -17.5 0 1 0 0 0 +20 1 1 1 0 0 0 -20 0 1 0 0 1 +22.5 1 1 0 0 1 -22.5 0 1 0 1 0 +25.0 1 1 0 1 0 -25.0 0 1 0 1 1 +27.5 1 1 0 1 1 -27.5 0 1 1 0 0 +30 1 1 1 0 0 -30 0 1 1 0 1 +32.5 1 1 1 0 1 -32.5 0 1 1 1 0 +35.0 1 1 1 1 0 -35.0 0 1 1 1 1 +37.5 1 1 1 1 1 -37.5
10	RSVD2	NOT_ACCESSIBLE	Reserved.
9-7	TWPRE	READ_WRITE	Precursor Tap Weight. Selects one of 8 output tap weights for TX waveform conditioning. The settings are from 0 to 17.5% in 2.5% steps. 0 0 0 0 0 0 1 -2.5 0 1 0 -5.0 0 1 1 -7.5 1 0 0 -10.0 1 0 1 -12.5 1 1 0 -15.0 1 1 1 -17.5
6	RSVD1	NOT_ACCESSIBLE	Reserved.

Table 8-14. SD_TX_R2_CFG[0] Register Field Descriptions (continued)

Bits	Field Name	Type	Description
5-2	SWING	READ_WRITE	Output swing. Selects one of 16 output amplitude settings between 795 and 1275 mVdfpp. If link termination has higher noise level, this should be set to maximum value. 0000 795mv 0001 830mv 0010 870mv 0011 905mv 0100 940mv 0101 975mv 0110 1010mv 0111 1045mv 1000 1080mv 1001 1110mv 1010 1145mv 1011 1175mv 1100 1200mv 1101 1230mv 1110 1255mv 1111 1275mv
1	RSVD	NOT_ACCESSIBLE	Reserved.
0	TXINVPAR	READ_WRITE	Invert Polarity. Inverts the polarity of TXPi and TXNi. 0 Normal polarity 1 Inverted polarity

SD_TX_R2_CFG[1]	**Access = READ_WRITE**	**Address [0x8818]**
SD_TX_R2_CFG[2]	**Access = READ_WRITE**	**Address [0x9018]**
SD_TX_R2_CFG[3]	**Access = READ_WRITE**	**Address [0x9818]**
SD_TX_R2_CFG[4]	**Access = READ_WRITE**	**Address [0xA018]**
SD_TX_R2_CFG[5]	**Access = READ_WRITE**	**Address [0xA818]**

8.2.1.8 SD_TX_STS[0] Register (offset = 0x801C)

Table 8-15. SD_TX_STS[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-5	RSVD1	NOT_ACCESSIBLE	Reserved.
4	TXTESTFAIL	READ	Test Failure. Driven high when an error is encountered during a test sequence executed on an individual channel. Synchronous to TXBCLK.
3	RSVD	NOT_ACCESSIBLE	Reserved.
2-0	TXBUSWIDTH	READ	The transmitter bus bandwidth is fixed in hardware to 20 bits and is not configurable through an MMR. However the value that reflects a 20-bit bus width will be read of the configuration register is read. 010 20-bit operation 011 16-bit operation
SD_TX_STS[1]		**Access = READ_WRITE**	**Address [0x881C]**
SD_TX_STS[2]		**Access = READ_WRITE**	**Address [0x901C]**
SD_TX_STS[3]		**Access = READ_WRITE**	**Address [0x981C]**
SD_TX_STS[4]		**Access = READ_WRITE**	**Address [0xA01C]**
SD_TX_STS[5]		**Access = READ_WRITE**	**Address [0xA81C]**

8.2.1.9 SD_PLL_B8_EN_CFG Register (offset = 0xB000)

Table 8-16. SD_PLL_B8_EN_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	ENPLL_B8	READ_WRITE	The transmit and receive SerDes functions requires that the PLL has been enabled. 0 = Disable B8 PLL 1 = Enable B8 PLL

8.2.1.10 SD_PLL_B4_EN_CFG Register (offset = 0xB004)

Table 8-17. SD_PLL_B4_EN_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	ENPLL_B4	READ_WRITE	The transmit and receive SerDes functions requires that the PLL has been enabled. 0 = Disable B4 PLL 1 = Enable B4 PLL

8.2.1.11 SD_PLL_B8_CFG Register (offset = 0xB008)

Table 8-18. SD_PLL_B8_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-15	RSVD1	NOT_ACCESSIBLE	Reserved.
14-13	CLKBYP_B8	READ_WRITE	Clock Bypass. Facilitates bypassing of the PLL with either REFCLKP/N or TESTCLKT, and bypassing of the recovered receiver clock with TESTCLKR. 00 No bypass 10 Functional bypass (The macro operates functionally at low speed using testclkt and testclkr) 11 Refclk Observe (The PLL is bypassed by refclkp/n)
12	RSVD	NOT_ACCESSIBLE	Reserved.
11-10	LB_B8	READ_WRITE	Loop Bandwidth. Specify loop bandwidth. 0 0 Medium Bandwidth. 0 1 Ultra High Bandwidth. 1 0 Low Bandwidth. 1 1 High Bandwidth.
9	SLEEPPLL_B8	READ_WRITE	Puts the B8 PLL into sleep state when high.
8	VRANGE_B8	READ_WRITE	Select between high and low range. 0 = Low Range 1 = High Range 0 = PLL awake 1 = PLL sleep
7-0	MPY_B8	READ_WRITE	The PLL of the SerDes module must be between 1.5625 Ghz and 3.125 Ghz for proper SerDes operation. The optimal reference clock frequency should be between 300 and 800 MHz, although a low frequency of 100 MHz can be used but at the expense of quality of the recovered and transmitted clock. Value Effect Value Effect 0 0 0 1 0 0 0 0 4x 0 0 1 1 0 0 1 0 12.5x 0 0 0 1 0 1 0 0 5x 0 0 1 1 1 1 0 0 15x 0 0 0 1 1 0 0 0 6x 0 1 0 0 0 0 0 0 16x 0 0 1 0 0 0 0 0 8x 0 1 0 0 0 0 1 0 16.5x 0 0 1 0 0 0 0 1 8.25x 0 1 0 1 0 0 0 0 20x 0 0 1 0 1 0 0 0 10x 0 1 0 1 1 0 0 0 22x 0 0 1 1 0 0 0 0 12x 0 1 1 0 0 1 0 0 25x

8.2.1.12 SD_PLL_B4_CFG Register (offset = 0xB00C)

Table 8-19. SD_PLL_B4_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-15	RSVD1	NOT_ACCESSIBLE	Reserved.
14-13	CLKBYP_B4	READ_WRITE	Clock Bypass. Facilitates bypassing of the PLL with either REFCLKP/N or TESTCLKT, and bypassing of the recovered receiver clock with TESTCLKR. 00 No bypass 10 Functional bypass (The macro operates functionally at low speed using testclkt and testclkr) 11 Refclk Observe (The PLL is bypassed by refclkp/n)
12	RSVD	NOT_ACCESSIBLE	Reserved.
11-10	LB_B4	READ_WRITE	Loop Bandwidth. Specify loop bandwidth. 0 0 Medium Bandwidth. 0 1 Ultra High Bandwidth. 1 0 Low Bandwidth. 1 1 High Bandwidth
9	SLEEPPLL_B4	READ_WRITE	Puts the B4 PLL into sleep state when high.
8	VRANGE_B4	READ_WRITE	Select between high and low range. 0 = Low Range 1 = High Range 0 = PLL awake 1 = PLL sleep
7-0	MPY_B4	READ_WRITE	The PLL of the SerDes module must be between 1.5625 Ghz and 3.125 Ghz for proper SerDes operation. The optimal reference clock frequency should be between 300 and 800 MHz, although a low frequency of 100 MHz can be used but at the expense of quality of the recovered and transmitted clock. Value Effect Value Effect 0 0 0 1 0 0 0 0 4x 0 0 1 1 0 0 1 0 12.5x 0 0 0 1 0 1 0 0 5x 0 0 1 1 1 1 0 0 15x 0 0 0 1 1 0 0 0 6x 0 1 0 0 0 0 0 0 16x 0 0 1 0 0 0 0 0 8x 0 1 0 0 0 0 1 0 16.5x 0 0 1 0 0 0 0 1 8.25x 0 1 0 1 0 0 0 0 20x 0 0 1 0 1 0 0 0 10x 0 1 0 1 1 0 0 0 22x 0 0 1 1 0 0 0 0 12x 0 1 1 0 0 1 0 0 25x

8.2.1.13 SD_PLL_B8_STS Register (offset = 0xB010)

Table 8-20. SD_PLL_B8_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	Lock_B8	READ	The transmitter and receiver SerDes macros require that the PLL has acquired lock to operate. The lock status should be validated before the SerDes macros are enabled.

8.2.1.14 SD_PLL_B4_STS Register (offset = 0xB014)

Table 8-21. SD_PLL_B4_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	Lock_B4	READ	The transmitter and receiver SerDes macros require that the PLL has acquired lock to operate. The lock status should be validated before the SerDes macros are enabled.

8.2.1.15 SD_CLK_SEL_CFG Register (offset = 0xB018)

Table 8-22. SD_CLK_SEL_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	clksel	READ_WRITE	The TX byte clock from either B8 or B4 SerDes link 0 will be selected as sys_clk once the PLL has acquired lock. 0 Select TX Byte clock 0 from B8 SerDes 1 Select TX Byte clock 0 from B4 SerDes

8.2.1.16 SD_LK_CLK_DIS_CFG Register (offset = 0xB01C)

Table 8-23. SD_LK_CLK_DIS_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-6	RSVD	NOT_ACCESSIBLE	Reserved.
5	DISCLK_LK5	READ_WRITE	The clock for this link will be gated off when set 0 = Enable Clock 1 = Disable Clock
4	DISCLK_LK4	READ_WRITE	The clock for this link will be gated off when set 0 = Enable Clock 1 = Disable Clock
3	DISCLK_LK3	READ_WRITE	The clock for this link will be gated off when set 0 = Enable Clock 1 = Disable Clock
2	DISCLK_LK2	READ_WRITE	The clock for this link will be gated off when set 0 = Enable Clock 1 = Disable Clock
1	DISCLK_LK1	READ_WRITE	The clock for this link will be gated off when set 0 = Enable Clock 1 = Disable Clock
0	DISCLK_LK0	READ_WRITE	The clock for this link will be gated off when set 0 = Enable Clock 1 = Disable Clock

8.3 RM Registers

Table 8-24. RM Register Memory Map

Offset	Register Name	Description	Section
0x50000	RM_LK_CFG0[0]	RM Link Configuration Register 0 (Link 0)	Section 8.3.1.1
0x50004	RM_LK_CFG1[0]	RM Link Configuration Register 1	Section 8.3.1.2
0x50008	RM_LK_CFG2[0]	RM Link Configuration Register 2	Section 8.3.1.3
0x5000C	RM_LK_CFG3[0]	RM Link Configuration Register 3	Section 8.3.1.4
0x50010	RM_LK_CFG4[0]	RM Link Configuration Register 4	Section 8.3.1.5
0x50014	RM_LK_STS0[0]	RM Link Status Register 0	Section 8.3.1.6
0x50018	RM_LK_STS1[0]	RM Link Status Register 1	Section 8.3.1.7
0x5001C	RM_LK_STS2[0]	RM Link Status Register 2	Section 8.3.1.8
0x50020	RM_LK_STS3[0]	RM Link Status Register 3	Section 8.3.1.9
0x50024	RM_LK_STS4[0]	RM Link Status Register 4	Section 8.3.1.10
0x50800	RM_LK_CFG0[1]	RM Link Configuration Register 0 (Link 1)	Section 8.3.1.1
0x50804	RM_LK_CFG1[1]	RM Link Configuration Register 1	Section 8.3.1.2
0x50808	RM_LK_CFG2[1]	RM Link Configuration Register 2	Section 8.3.1.3
0x5080C	RM_LK_CFG3[1]	RM Link Configuration Register 3	Section 8.3.1.4
0x50810	RM_LK_CFG4[1]	RM Link Configuration Register 4	Section 8.3.1.5
0x50814	RM_LK_STS0[1]	RM Link Status Register 0	Section 8.3.1.6
0x50818	RM_LK_STS1[1]	RM Link Status Register 1	Section 8.3.1.7
0x5081C	RM_LK_STS2[1]	RM Link Status Register 2	Section 8.3.1.8
0x50820	RM_LK_STS3[1]	RM Link Status Register 3	Section 8.3.1.9
0x50824	RM_LK_STS4[1]	RM Link Status Register 4	Section 8.3.1.10
0x51000	RM_LK_CFG0[2]	RM Link Configuration Register 0 (Link 2)	Section 8.3.1.1
0x51004	RM_LK_CFG1[2]	RM Link Configuration Register 1	Section 8.3.1.2
0x51008	RM_LK_CFG2[2]	RM Link Configuration Register 2	Section 8.3.1.3
0x5100C	RM_LK_CFG3[2]	RM Link Configuration Register 3	Section 8.3.1.4
0x51010	RM_LK_CFG4[2]	RM Link Configuration Register 4	Section 8.3.1.5
0x51014	RM_LK_STS0[2]	RM Link Status Register 0	Section 8.3.1.6
0x51018	RM_LK_STS1[2]	RM Link Status Register 1	Section 8.3.1.7
0x5101C	RM_LK_STS2[2]	RM Link Status Register 2	Section 8.3.1.8
0x51020	RM_LK_STS3[2]	RM Link Status Register 3	Section 8.3.1.9
0x51024	RM_LK_STS4[2]	RM Link Status Register 4	Section 8.3.1.10
0x51800	RM_LK_CFG0[3]	RM Link Configuration Register 0 (Link 3)	Section 8.3.1.1
0x51804	RM_LK_CFG1[3]	RM Link Configuration Register 1	Section 8.3.1.2
0x51808	RM_LK_CFG2[3]	RM Link Configuration Register 2	Section 8.3.1.3
0x5180C	RM_LK_CFG3[3]	RM Link Configuration Register 3	Section 8.3.1.4
0x51810	RM_LK_CFG4[3]	RM Link Configuration Register 4	Section 8.3.1.5
0x51814	RM_LK_STS0[3]	RM Link Status Register 0	Section 8.3.1.6
0x51818	RM_LK_STS1[3]	RM Link Status Register 1	Section 8.3.1.7
0x5181C	RM_LK_STS2[3]	RM Link Status Register 2	Section 8.3.1.8
0x51820	RM_LK_STS3[3]	RM Link Status Register 3	Section 8.3.1.9
0x51824	RM_LK_STS4[3]	RM Link Status Register 4	Section 8.3.1.10
0x52000	RM_LK_CFG0[4]	RM Link Configuration Register 0 (Link 4)	Section 8.3.1.1
0x52004	RM_LK_CFG1[4]	RM Link Configuration Register 1	Section 8.3.1.2
0x52008	RM_LK_CFG2[4]	RM Link Configuration Register 2	Section 8.3.1.3
0x5200C	RM_LK_CFG3[4]	RM Link Configuration Register 3	Section 8.3.1.4
0x52010	RM_LK_CFG4[4]	RM Link Configuration Register 4	Section 8.3.1.5

Table 8-24. RM Register Memory Map (continued)

Offset	Register Name	Description	Section
0x52014	RM_LK_STS0[4]	RM Link Status Register 0	Section 8.3.1.6
0x52018	RM_LK_STS1[4]	RM Link Status Register 1	Section 8.3.1.7
0x5201C	RM_LK_STS2[4]	RM Link Status Register 2	Section 8.3.1.8
0x52020	RM_LK_STS3[4]	RM Link Status Register 3	Section 8.3.1.9
0x52024	RM_LK_STS4[4]	RM Link Status Register 4	Section 8.3.1.10
0x52800	RM_LK_CFG0[5]	RM Link Configuration Register 0 (Link 5)	Section 8.3.1.1
0x52804	RM_LK_CFG1[5]	RM Link Configuration Register 1	Section 8.3.1.2
0x52808	RM_LK_CFG2[5]	RM Link Configuration Register 2	Section 8.3.1.3
0x5280C	RM_LK_CFG3[5]	RM Link Configuration Register 3	Section 8.3.1.4
0x52810	RM_LK_CFG4[5]	RM Link Configuration Register 4	Section 8.3.1.5
0x52814	RM_LK_STS0[5]	RM Link Status Register 0	Section 8.3.1.6
0x52818	RM_LK_STS1[5]	RM Link Status Register 1	Section 8.3.1.7
0x5281C	RM_LK_STS2[5]	RM Link Status Register 2	Section 8.3.1.8
0x52820	RM_LK_STS3[5]	RM Link Status Register 3	Section 8.3.1.9
0x52824	RM_LK_STS4[5]	RM Link Status Register 4	Section 8.3.1.10
0x53000	RM_CFG	RM Configuration Register	Section 8.3.1.11

8.3.1 Link Registers Details

8.3.1.1 RM_LK_CFG0[0] Register (offset = 0x50000)

Table 8-25. RM_LK_CFG0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	RSVD1	NOT_ACCESSIBLE	RESERVED
15	lcv_cntr_en	READ_WRITE	Writing a 1 to the bit will enable the Line Code Violation counter. THis 16-bit counter will saturate when it reaches a value of 0xffff. Writing a 0 to this bit will clear and disable the counter. The current counter value is available as status, lcv_cntr_value 0 = lcv_cntr disabled and cleared to a value of 0x0000. 1 = lcv_cntr enabled, counts each LCV to a max of 0xffff
14	lcv_unsync_en	READ_WRITE	Enables a state transition from the ST3 to the ST0 state when lcv_det_thold is met 0 = lcv_det_thold has no effect on the RX FSM 1 = The RX FSM transitions from ST3 to ST0 when lcv_det_thold is met
13	scr_en	READ_WRITE	Enables the scrambler for OBSAI 8x link rate operation in the receiver data path 0 = RM scrambler Disabled 1 = RM scrambler Enabled
12	sd_auto_align_en	READ_WRITE	Enables the RM to automatically disable Serdes symbol alignment when the receiver state machine reaches state ST3 0 = Disable auto alignment 1 = Enable auto alignment
11	error_suppress	READ_WRITE	Suppress error reporting when the receiver state machine is not in state ST3 0 = Allow all RM error reporting when not in ST3 1 = Suppress all RM error reporting when not in ST3
10-8	force_rx_state	READ_WRITE	Force receiver state machine state 2 = Force ST4 state 3 = Force ST5 state 4 = Force ST0 state 5 = Force ST1 state 6 = Force ST2 state 7 = Force ST3 state
7-6	RSVD	NOT_ACCESSIBLE	RESERVED
5-4	fifo_thold	READ_WRITE	Sets the watermark of where the RM begins reading from the RX FIFO 0 = FIFO starts reading immediately 1 = FIFO starts reading after 4 dual words received 2 = FIFO starts reading after 8 dual words received 3 = FIFO starts reading after 16 dual words received
3-2	link_rate	READ_WRITE	The link rate field defines the rate of each link that each block 0 = 8x rate 1 = 4x rate 2 = 2x rate 3 = 5x rate, CPRI only
1	rx_en	READ_WRITE	Enable RM Link 0 = RM link disable 1 = RM link enable
0	mode_sel	READ_WRITE	Enables short frame mode for all six links 0 = Select CPRI operation mode 1 = Select OBSAI operation mode
RM_LK_CFG0[1]			**Address [0x50800]**
RM_LK_CFG0[2]			**Address [0x51000]**
RM_LK_CFG0[3]			**Address [0x51800]**
RM_LK_CFG0[4]			**Address [0x52000]**
RM_LK_CFG0[5]			**Address [0x52800]**

8.3.1.2 RM_LK_CFG1[0] Register (offset = 0x50004)

Table 8-26. RM_LK_CFG1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	mon_wrap	READ_WRITE	Defines the wrap value of the clock monitor used to define clock quality. A value of zero disables the clock monitor, Range 0 to 65,535
15-10	RSVD	NOT_ACCESSIBLE	RESERVED
9	cq_en	READ_WRITE	Enables the clock quality circuit. 0 = Clock quality circuit Disabled. 1 = Clock quality circuit Enabled.
8	wd_en	READ_WRITE	Enables the clock detect watch dog timer. 0 = Clock detect watch dog timer Disabled. 1 = Clock detect watch dog timer Enabled.
7-0	wd_wrap	READ_WRITE	Defines the wrap value of the clock detection watchdog circuit. A value of zero disables the clock watchdog timer, Range 0 to 255
RM_LK_CFG1[1]			**Address [0x50804]**
RM_LK_CFG1[2]			**Address [0x51004]**
RM_LK_CFG1[3]			**Address [0x51804]**
RM_LK_CFG1[4]			**Address [0x52004]**
RM_LK_CFG1[5]			**Address [0x52804]**

8.3.1.3 RM_LK_CFG2[0] Register (offset = 0x50008)

Table 8-27. RM_LK_CFG2[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	RSVD	NOT_ACCESSIBLE	RESERVED
15-0	los_det_thold	READ_WRITE	Sets 8b10b los detect threshold values in number of Line Code Violations received during a master frame, OBSAI, or during a Hyperframe, CPRI. Writing to this location will automatically clear the num_los counter and num_los_det status bit. Range 0 to 65,535
RM_LK_CFG2[1]			**Address [0x50808]**
RM_LK_CFG2[2]			**Address [0x51008]**
RM_LK_CFG2[3]			**Address [0x51808]**
RM_LK_CFG2[4]			**Address [0x52008]**
RM_LK_CFG2[5]			**Address [0x52808]**

8.3.1.4 RM_LK_CFG3[0] Register (offset = 0x5000C)

Table 8-28. RM_LK_CFG3[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	frame_sync_t	READ_WRITE	Threshold value for consecutive valid message groups that result in state ST3. Range 0 to 65,535
15-0	sync_t	READ_WRITE	Threshold value for consecutive valid blocks of bytes that result in state ST1. Range 0 to 65,535
RM_LK_CFG3[1]			**Address [0x5080C]**
RM_LK_CFG3[2]			**Address [0x5100C]**
RM_LK_CFG3[3]			**Address [0x5180C]**
RM_LK_CFG3[4]			**Address [0x5200C]**
RM_LK_CFG3[5]			**Address [0x5280C]**

8.3.1.5 RM_LK_CFG4[0] Register (offset = 0x50010)

Table 8-29. RM_LK_CFG4[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	frame_unsync_t	READ_WRITE	Threshold value for consecutive invalid message groups that result in state ST1. Range 0 to 65,535
15-0	unsync_t	READ_WRITE	Threshold value for consecutive invalid blocks of bytes that result in state ST0. Range 0 to 65,535
RM_LK_CFG4[1]			**Address [0x50810]**
RM_LK_CFG4[2]			**Address [0x51010]**
RM_LK_CFG4[3]			**Address [0x51810]**
RM_LK_CFG4[4]			**Address [0x52010]**
RM_LK_CFG4[5]			**Address [0x52810]**

8.3.1.6 RM_LK_STS0[0] Register (offset = 0x50014)

Table 8-30. RM_LK_STS0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD2	NOT_ACCESSIBLE	RESERVED
11	fifo_ovf	READ	Active after an RM FIFO overflow 0 = No FIFO error 1 = FIFO Overflow occurred
10	loc	READ	The clock watchdog circuit detected a missing clock 0 = RX clock watchdog not enabled or not missing a clock 1 = RX clock watchdog detected a missing a clock
9	num_los_det	READ	Detects when num_los counter has reached the program able los_det_thold limit 0 = After a master frame, in OBSAI, or Hyperframe, in CPRI, of no 8b10b errors or when configuring value los_det_thold is written 1 = When num_los counter has reached los_det_thold within a master frame, OBSAI, or Hyperframe, CPRI.
8	los	READ	Active when RX state machine is in ST0, inactive otherwise 0 = RX FSM Not in ST0 State 1 = RX FSM is in ST0 State
7-6	RSVD1	NOT_ACCESSIBLE	RESERVED
5-0	sync_status	READ	Indicates the current status of the RX state machine 8 = ST0 State UNSYNC 4 = ST1 State WAIT_FOR_K28p7_IDLES 2 = ST2 State WAIT_FOR_FRAME_SYNC_T 1 = ST3 State FRAME_SYNC 16 = ST4 State WAIT_FOR_SEED 32 = ST5 State WAIT_FOR_ACK
RM_LK_STS0[1]			**Address [0x50814]**
RM_LK_STS0[2]			**Address [0x51014]**
RM_LK_STS0[3]			**Address [0x51814]**
RM_LK_STS0[4]			**Address [0x52014]**
RM_LK_STS0[5]			**Address [0x52814]**

8.3.1.7 RM_LK_STS1[0] Register (offset = 0x50018)

Table 8-31. RM_LK_STS1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	lcv_cntr_value	READ	Number of Line Code Violations counted since last cleared and enabled. Range 0 to 65,535
15-0	num_los	READ	Represents the number of los_det, 8b10b code violation, occurrences in a master frame, in OBSAI, or hyperframe, in CPRI. This counter will saturate and hold its value until either cleared by writing the configuration value los_det_thold = 0 or after a master frame, in OBSAI, or hyperframe, in CPRI of no 8b10b errors. Range 0 to 65,535
RM_LK_STS1[1]			**Address [0x50818]**
RM_LK_STS1[2]			**Address [0x51018]**
RM_LK_STS1[3]			**Address [0x51818]**
RM_LK_STS1[4]			**Address [0x52018]**
RM_LK_STS1[5]			**Address [0x52818]**

8.3.1.8 RM_LK_STS2[0] Register (offset = 0x5001C)

Table 8-32. RM_LK_STS2[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-23	RSVD	NOT_ACCESSIBLE	RESERVED
22-16	scr_value	READ	Indicates the captured scrambling code, only when configuration bit scr_en = 1.
15-0	clk_qual	READ	A value that represents the quality, relative frequency, of the received Serdes clock. Range 0 to 65,535
RM_LK_STS2[1]			**Address [0x5081C]**
RM_LK_STS2[2]			**Address [0x5101C]**
RM_LK_STS2[3]			**Address [0x5181C]**
RM_LK_STS2[4]			**Address [0x5201C]**
RM_LK_STS2[5]			**Address [0x5281C]**

8.3.1.9 RM_LK_STS3[0] Register (offset = 0x50020)

Table 8-33. RM_LK_STS3[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD	NOT_ACCESSIBLE	RESERVED
25	lof_state	READ	Active high status indicates Loss Of Frame when the receiver FSM is in state ST0 or ST1. NOTE: The value of this bit will be 0 after reset but will change to a value of 1 if CPRI mode is enabled.
24	hfsync_state	READ	Active high status indicates when the receiver FSM is in the HFSYNC state ST3
23-16	bfn_high	READ	Received Node B Frame number high byte, Z.130.0
15-8	bfn_low	READ	Received Node B Frame number low byte, Z.128.0
7-0	hfn	READ	Received hyperframe number, Z.64.0. Range 0 to 149 basic frames
RM_LK_STS3[1]			**Address [0x50820]**
RM_LK_STS3[2]			**Address [0x51020]**
RM_LK_STS3[3]			**Address [0x51820]**
RM_LK_STS3[4]			**Address [0x52020]**
RM_LK_STS3[5]			**Address [0x52820]**

8.3.1.10 RM_LK_STS4[0] Register (offset = 0x50024)

Table 8-34. RM_LK_STS4[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-24	l1_pntr_p	READ	Received Pointer P
23-21	RSVD	NOT_ACCESSIBLE	RESERVED
20	l1_rcvd_lof	READ	Received sdi, Z.130.0, b4
19	l1_rcvd_los	READ	Received sdi, Z.130.0, b3
18	l1_rcvd_sdi	READ	Received sdi, Z.130.0, b2
17	l1_rcvd_rai	READ	Received rai, Z.130.0, b1
16	l1_rcvd_rst	READ	Received reset, Z.130.0, b0
15-8	l1_start_up	READ	Received start up information, Z.66.0
7-0	l1_version	READ	Received protocol version, Z.2.0
RM_LK_STS4[1]			**Address [0x50824]**
RM_LK_STS4[2]			**Address [0x51024]**
RM_LK_STS4[3]			**Address [0x51824]**
RM_LK_STS4[4]			**Address [0x52024]**
RM_LK_STS4[5]			**Address [0x52824]**

8.3.1.11 RM_CFG Register (offset = 0x53000)

Table 8-35. RM_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD1	NOT_ACCESSIBLE	RESERVED
8	short_frm_en	READ_WRITE	Short Frame enable 0 = Long frame mode 1 = Short frame mode
7-3	RSVD	NOT_ACCESSIBLE	RESERVED
2-0	raw_data_sel	READ_WRITE	Select 1 of 6 links of raw, not gated, RM output data used for data trace feature 0 = Link 0 Selected 1 = Link 1 Selected 2 = Link 2 Selected 3 = Link 3 Selected 4 = Link 4 Selected 5 = Link 5 Selected

8.4 TM Registers

Table 8-36. TM Register Memory Map

Offset	Register Name	Description	Section
0x4C000	TM_LK_CFG[0]	The TM Configuration Register is used to program basic functionality of the Tx Mac Block. Bit 7 can be updated at any time to turn on or off the link, all other bits should only be updated if Bit 7 is 0	Section 8.4.1.1
0x4C004	TM_LK_CTRL[0]	TM State Machine Control Register	Section 8.4.1.2
0x4C008	TM_LK_SCR_CTRL[0]	The Scrambler Configuration Register contains the seed initialization vector for the LFSR scrambler utilized when scrambling is enabled is OBSAI 8x mode, and the scrambler enable bit. This configuration register should only be updated when the Frame Sync state machine is disabled.	Section 8.4.1.3
0x4C00C	TM_LK_L1_CFG[0]	Programs L1 inband control signals for CPRI mode	Section 8.4.1.4
0x4C010	TM_LK_L1_EN[0]	The L1 Inband Enable Register allows hardware control of the L1 inband control signals for CPRI mode. A 1 for each bit indicates the hardware control is enabled. Each enable bit is a gate on an input condition that could affect an output condition. The nomenclature is defined as TXSIGNAL_RXCOND_EN. The term ERR indicates the error had been determined by the RM, while the term RX refers to the actual L1 inband signal received by the RM.	Section 8.4.1.5
0x4C014	TM_LK_LOSERR[0]	Selects which RM link is used to drive the LOSERR condition to determine transmit L1 Inband signaling	Section 8.4.1.6
0x4C018	TM_LK_LOFERR[0]	Selects which RM link is used to drive the LOFERR condition to determine transmit L1 Inband signaling	Section 8.4.1.7
0x4C01C	TM_LK_LOSRx[0]	Selects which RM link is used to drive the LOSRx condition to determine transmit L1 Inband signaling	Section 8.4.1.8
0x4C020	TM_LK_LOFRx[0]	Selects which RM link is used to drive the LOFRx condition to determine transmit L1 Inband signaling	Section 8.4.1.9
0x4C024	TM_LK_RAIRx[0]	Selects which RM link is used to drive the RAIRx condition to determine transmit L1 Inband signaling	Section 8.4.1.10
0x4C028	TM_LK_HFN[0]	TM CPRI HFN Status	Section 8.4.1.11
0x4C02C	TM_LK_PTRP[0]	Contains Pointer P value	Section 8.4.1.12
0x4C030	TM_LK_STRT[0]	Contains Startup value	Section 8.4.1.13
0x4C034	TM_LK_PROT[0]	Contains Protocol Version value	Section 8.4.1.14
0x4C038	TM_LK_STAT[0]	The TM Status Register contains status of the TM block	Section 8.4.1.15
0x4C03C	TM_FRM_MODE[0]	Enables Short Frame Mode	Section 8.4.1.16
0x4C800	TM_LK_CFG[1]	The TM Configuration Register is used to program basic functionality of the Tx Mac Block. Bit 7 can be updated at any time to turn on or off the link, all other bits should only be updated if Bit 7 is 0	Section 8.4.1.1
0x4C804	TM_LK_CTRL[1]	TM State Machine Control Register	Section 8.4.1.2
0x4C808	TM_LK_SCR_CTRL[1]	The Scrambler Configuration Register contains the seed initialization vector for the LFSR scrambler utilized when scrambling is enabled is OBSAI 8x mode, and the scrambler enable bit. This configuration register should only be updated when the Frame Sync state machine is disabled.	Section 8.4.1.3
0x4C80C	TM_LK_L1_CFG[1]	Programs L1 inband control signals for CPRI mode	Section 8.4.1.4
0x4C810	TM_LK_L1_EN[1]	The L1 Inband Enable Register allows hardware control of the L1 inband control signals for CPRI mode. A 1 for each bit indicates the hardware control is enabled. Each enable bit is a gate on an input condition that could affect an output condition. The nomenclature is defined as TXSIGNAL_RXCOND_EN. The term ERR indicates the error had been determined by the RM, while the term RX refers to the actual L1 inband signal received by the RM.	Section 8.4.1.5
0x4C814	TM_LK_LOSERR[1]	Selects which RM link is used to drive the LOSERR condition to determine transmit L1 Inband signaling	Section 8.4.1.6

Table 8-36. TM Register Memory Map (continued)

Offset	Register Name	Description	Section
0x4C818	TM_LK_LOFERR[1]	Selects which RM link is used to drive the LORERR condition to determine transmit L1 Inband signaling	Section 8.4.1.7
0x4C81C	TM_LK_LOSRx[1]	Selects which RM link is used to drive the LOSRx condition to determine transmit L1 Inband signaling	Section 8.4.1.8
0x4C820	TM_LK_LOFRx[1]	Selects which RM link is used to drive the LOFRx condition to determine transmit L1 Inband signaling	Section 8.4.1.9
0x4C824	TM_LK_RAIRx[1]	Selects which RM link is used to drive the RAIRx condition to determine transmit L1 Inband signaling	Section 8.4.1.10
0x4C828	TM_LK_HFN[1]	TM CPRI HFN Status	Section 8.4.1.11
0x4C82C	TM_LK_PTRP[1]	Contains Pointer P value	Section 8.4.1.12
0x4C830	TM_LK_STRT[1]	Contains Startup value	Section 8.4.1.13
0x4C834	TM_LK_PROT[1]	Contains Protocol Version value	Section 8.4.1.14
0x4C838	TM_LK_STAT[1]	The TM Status Register contains status of the TM block	Section 8.4.1.15
0x4C83C	TM_FRM_MODE[1]	Enables Short Frame Mode	Section 8.4.1.16
0x4D000	TM_LK_CFG[2]	The TM Configuration Register is used to program basic functionality of the Tx Mac Block. Bit 7 can be updated at any time to turn on or off the link, all other bits should only be updated if Bit 7 is 0	Section 8.4.1.1
0x4D004	TM_LK_CTRL[2]	TM State Machine Control Register	Section 8.4.1.2
0x4D008	TM_LK_SCR_CTRL[2]	The Scrambler Configuration Register contains the seed initialization vector for the LFSR scrambler utilized when scrambling is enabled is OBSAI 8x mode, and the scrambler enable bit. This configuration register should only be updated when the Frame Sync state machine is disabled.	Section 8.4.1.3
0x4D00C	TM_LK_L1_CFG[2]	Programs L1 inband control signals for CPRI mode	Section 8.4.1.4
0x4D010	TM_LK_L1_EN[2]	The L1 Inband Enable Register allows hardware control of the L1 inband control signals for CPRI mode. A 1 for each bit indicates the hardware control is enabled. Each enable bit is a gate on an input condition that could affect an output condition. The nomenclature is defined as TXSIGNAL_RXCOND_EN. The term ERR indicates the error had been determined by the RM, while the term RX refers to the actual L1 inband signal received by the RM.	Section 8.4.1.5
0x4D014	TM_LK_LOSERR[2]	Selects which RM link is used to drive the LOSERR condition to determine transmit L1 Inband signaling	Section 8.4.1.6
0x4D018	TM_LK_LOFERR[2]	Selects which RM link is used to drive the LORERR condition to determine transmit L1 Inband signaling	Section 8.4.1.7
0x4D01C	TM_LK_LOSRx[2]	Selects which RM link is used to drive the LOSRx condition to determine transmit L1 Inband signaling	Section 8.4.1.8
0x4D020	TM_LK_LOFRx[2]	Selects which RM link is used to drive the LOFRx condition to determine transmit L1 Inband signaling	Section 8.4.1.9
0x4D024	TM_LK_RAIRx[2]	Selects which RM link is used to drive the RAIRx condition to determine transmit L1 Inband signaling	Section 8.4.1.10
0x4D028	TM_LK_HFN[2]	TM CPRI HFN Status	Section 8.4.1.11
0x4D02C	TM_LK_PTRP[2]	Contains Pointer P value	Section 8.4.1.12
0x4D030	TM_LK_STRT[2]	Contains Startup value	Section 8.4.1.13
0x4D034	TM_LK_PROT[2]	Contains Protocol Version value	Section 8.4.1.14
0x4D038	TM_LK_STAT[2]	The TM Status Register contains status of the TM block	Section 8.4.1.15
0x4D03C	TM_FRM_MODE[2]	Enables Short Frame Mode	Section 8.4.1.16
0x4D800	TM_LK_CFG[3]	The TM Configuration Register is used to program basic functionality of the Tx Mac Block. Bit 7 can be updated at any time to turn on or off the link, all other bits should only be updated if Bit 7 is 0	Section 8.4.1.1
0x4D804	TM_LK_CTRL[3]	TM State Machine Control Register	Section 8.4.1.2

Table 8-36. TM Register Memory Map (continued)

Offset	Register Name	Description	Section
0x4D808	TM_LK_SCR_CTRL[3]	The Scrambler Configuration Register contains the seed initialization vector for the LFSR scrambler utilized when scrambling is enabled is OBSAI 8x mode, and the scrambler enable bit. This configuration register should only be updated when the Frame Sync state machine is disabled.	Section 8.4.1.3
0x4D80C	TM_LK_L1_CFG[3]	Programs L1 inband control signals for CPRI mode	Section 8.4.1.4
0x4D810	TM_LK_L1_EN[3]	The L1 Inband Enable Register allows hardware control of the L1 inband control signals for CPRI mode. A 1 for each bit indicates the hardware control is enabled. Each enable bit is a gate on an input condition that could affect an output condition. The nomenclature is defined as TXSIGNAL_RXCOND_EN. The term ERR indicates the error had been determined by the RM, while the term RX refers to the actual L1 inband signal received by the RM.	Section 8.4.1.5
0x4D814	TM_LK_LOSERR[3]	Selects which RM link is used to drive the LOSERR condition to determine transmit L1 Inband signaling	Section 8.4.1.6
0x4D818	TM_LK_LOFERR[3]	Selects which RM link is used to drive the LORERR condition to determine transmit L1 Inband signaling	Section 8.4.1.7
0x4D81C	TM_LK_LOSRx[3]	Selects which RM link is used to drive the LOSRx condition to determine transmit L1 Inband signaling	Section 8.4.1.8
0x4D820	TM_LK_LOFRx[3]	Selects which RM link is used to drive the LOFRx condition to determine transmit L1 Inband signaling	Section 8.4.1.9
0x4D824	TM_LK_RAIRx[3]	Selects which RM link is used to drive the RAIRx condition to determine transmit L1 Inband signaling	Section 8.4.1.10
0x4D828	TM_LK_HFN[3]	TM CPRI HFN Status	Section 8.4.1.11
0x4D82C	TM_LK_PTRP[3]	Contains Pointer P value	Section 8.4.1.12
0x4D830	TM_LK_STRT[3]	Contains Startup value	Section 8.4.1.13
0x4D834	TM_LK_PROT[3]	Contains Protocol Version value	Section 8.4.1.14
0x4D838	TM_LK_STAT[3]	The TM Status Register contains status of the TM block	Section 8.4.1.15
0x4D83C	TM_FRM_MODE[3]	Enables Short Frame Mode	Section 8.4.1.16
0x4E000	TM_LK_CFG[4]	The TM Configuration Register is used to program basic functionality of the Tx Mac Block. Bit 7 can be updated at any time to turn on or off the link, all other bits should only be updated if Bit 7 is 0	Section 8.4.1.1
0x4E004	TM_LK_CTRL[4]	TM State Machine Control Register	Section 8.4.1.2
0x4E008	TM_LK_SCR_CTRL[4]	The Scrambler Configuration Register contains the seed initialization vector for the LFSR scrambler utilized when scrambling is enabled is OBSAI 8x mode, and the scrambler enable bit. This configuration register should only be updated when the Frame Sync state machine is disabled.	Section 8.4.1.3
0x4E00C	TM_LK_L1_CFG[4]	Programs L1 inband control signals for CPRI mode	Section 8.4.1.4
0x4E010	TM_LK_L1_EN[4]	The L1 Inband Enable Register allows hardware control of the L1 inband control signals for CPRI mode. A 1 for each bit indicates the hardware control is enabled. Each enable bit is a gate on an input condition that could affect an output condition. The nomenclature is defined as TXSIGNAL_RXCOND_EN. The term ERR indicates the error had been determined by the RM, while the term RX refers to the actual L1 inband signal received by the RM.	Section 8.4.1.5
0x4E014	TM_LK_LOSERR[4]	Selects which RM link is used to drive the LOSERR condition to determine transmit L1 Inband signaling	Section 8.4.1.6
0x4E018	TM_LK_LOFERR[4]	Selects which RM link is used to drive the LORERR condition to determine transmit L1 Inband signaling	Section 8.4.1.7
0x4E01C	TM_LK_LOSRx[4]	Selects which RM link is used to drive the LOSRx condition to determine transmit L1 Inband signaling	Section 8.4.1.8
0x4E020	TM_LK_LOFRx[4]	Selects which RM link is used to drive the LOFRx condition to determine transmit L1 Inband signaling	Section 8.4.1.9

Table 8-36. TM Register Memory Map (continued)

Offset	Register Name	Description	Section
0x4E024	TM_LK_RAIRx[4]	Selects which RM link is used to drive the RAIRx condition to determine transmit L1 Inband signaling	Section 8.4.1.10
0x4E028	TM_LK_HFN[4]	TM CPRI HFN Status	Section 8.4.1.11
0x4E02C	TM_LK_PTRP[4]	Contains Pointer P value	Section 8.4.1.12
0x4E030	TM_LK_STRT[4]	Contains Startup value	Section 8.4.1.13
0x4E034	TM_LK_PROT[4]	Contains Protocol Version value	Section 8.4.1.14
0x4E038	TM_LK_STAT[4]	The TM Status Register contains status of the TM block	Section 8.4.1.15
0x4E03C	TM_FRM_MODE[4]	Enables Short Frame Mode	Section 8.4.1.16
0x4E800	TM_LK_CFG[5]	The TM Configuration Register is used to program basic functionality of the Tx Mac Block. Bit 7 can be updated at any time to turn on or off the link, all other bits should only be updated if Bit 7 is 0	Section 8.4.1.1
0x4E804	TM_LK_CTRL[5]	TM State Machine Control Register	Section 8.4.1.2
0x4E808	TM_LK_SCR_CTRL[5]	The Scrambler Configuration Register contains the seed initialization vector for the LFSR scrambler utilized when scrambling is enabled is OBSAI 8x mode, and the scrambler enable bit. This configuration register should only be updated when the Frame Sync state machine is disabled.	Section 8.4.1.3
0x4E80C	TM_LK_L1_CFG[5]	Programs L1 inband control signals for CPRI mode	Section 8.4.1.4
0x4E810	TM_LK_L1_EN[5]	The L1 Inband Enable Register allows hardware control of the L1 inband control signals for CPRI mode. A 1 for each bit indicates the hardware control is enabled. Each enable bit is a gate on an input condition that could affect an output condition. The nomenclature is defined as TXSIGNAL_RXCOND_EN. The term ERR indicates the error had been determined by the RM, while the term RX refers to the actual L1 inband signal received by the RM.	Section 8.4.1.5
0x4E814	TM_LK_LOSERR[5]	Selects which RM link is used to drive the LOSERR condition to determine transmit L1 Inband signaling	Section 8.4.1.6
0x4E818	TM_LK_LOFERR[5]	Selects which RM link is used to drive the LORERR condition to determine transmit L1 Inband signaling	Section 8.4.1.7
0x4E81C	TM_LK_LOSRx[5]	Selects which RM link is used to drive the LOSRx condition to determine transmit L1 Inband signaling	Section 8.4.1.8
0x4E820	TM_LK_LOFRx[5]	Selects which RM link is used to drive the LOFRx condition to determine transmit L1 Inband signaling	Section 8.4.1.9
0x4E824	TM_LK_RAIRx[5]	Selects which RM link is used to drive the RAIRx condition to determine transmit L1 Inband signaling	Section 8.4.1.10
0x4E828	TM_LK_HFN[5]	TM CPRI HFN Status	Section 8.4.1.11
0x4E82C	TM_LK_PTRP[5]	Contains Pointer P value	Section 8.4.1.12
0x4E830	TM_LK_STRT[5]	Contains Startup value	Section 8.4.1.13
0x4E834	TM_LK_PROT[5]	Contains Protocol Version value	Section 8.4.1.14
0x4E838	TM_LK_STAT[5]	The TM Status Register contains status of the TM block	Section 8.4.1.15
0x4E83C	TM_FRM_MODE[5]	Enables Short Frame Mode	Section 8.4.1.16

8.4.1 Link Registers Details

8.4.1.1 TM_LK_CFG[0] Register (offset = 0x4C000)

Table 8-37. TM_LK_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD1	NOT_ACCESSIBLE	Reserved
7	TM_EN	READ_WRITE	The Transmit Enable Bit allows the TM block transmit state machine to operate. The following configurations 0 = TM Block Disabled 1 = TM Block Enabled
6-3	RSVD	NOT_ACCESSIBLE	Reserved
2	OBSAI_CPRI	READ_WRITE	The OBSAI/CPRI bit defines the mode of operation for the block. This bit should only be updated if TM Enable is 0 0 = CPRI Mode 1 = OBSAI Mode
1-0	Link_Rate	READ_WRITE	The link rate field defines the rate of each link that each block will provide conversion. This field should only be updated if TM Enable is 0. The field can be programmed for the following rates 0 = 8x Rate 1 = 4x Rate 2 = 2x Rate 3 = 5x Rate
TM_LK_CFG[1]			**Address [0x4C800]**
TM_LK_CFG[2]			**Address [0x4D000]**
TM_LK_CFG[3]			**Address [0x4D800]**
TM_LK_CFG[4]			**Address [0x4E000]**
TM_LK_CFG[5]			**Address [0x4E800]**

8.4.1.2 TM_LK_CTRL[0] Register (offset = 0x4C004)

Table 8-38. TM_LK_CTRL[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved
3	LOS_EN	READ_WRITE	Allows Loss of Signal from the RM to cause the TM state machine to transition to the OFF state. 0 = Loss of Signal Disabled 1 = Loss of Signal Enabled
2	TM_RESYNC	READ_WRITE	Forces the Frame Sync state machine to transition from the FRAME_SYNC state to the RESYNC state and remain in the RESYNC state until the bit is cleared. If the state machine is in either the OFF state or the IDLE state, this bit is ignored until the state machine reaches the RESYNC state 0 = No Action 1 = Forces the Frame Sync state machine to transition from the FRAME_SYNC state to the RESYNC state and remain in the RESYNC state until the bit is cleared
1	TM_Idle	READ_WRITE	Forces the Frame Sync state machine to transition from either the FRAME_SYNC or RESYNC states to the IDLE state and remain in the IDLE state until the bit is cleared. If the state machine is in the OFF state, this bit is ignored until the state machine reaches the IDLE state 0 = No Action 1 = Force to Idle state and remain until cleared
0	TM_Flush	READ_WRITE	Instructs the TM link to flush the FIFO and force a TM Fail condition on the link. When set high, the State machine is forced into the RE_SYNC state 0 = No Action 1 = Flush FIFO and force TM Fail condition
TM_LK_CTRL[1]			**Address [0x4C804]**
TM_LK_CTRL[2]			**Address [0x4D004]**
TM_LK_CTRL[3]			**Address [0x4D804]**
TM_LK_CTRL[4]			**Address [0x4E004]**
TM_LK_CTRL[5]			**Address [0x4E804]**

8.4.1.3 TM_LK_SCR_CTRL[0] Register (offset = 0x4C008)

Table 8-39. TM_LK_SCR_CTRL[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved
7	SCR_EN	READ_WRITE	Enables the scrambler for OBSAI 8x link rate operation in the transmit data path. 0 = Scrambler Disabled 1 = Scrambler Enabled
6-0	Seed_Value	READ_WRITE	The seed value is used to initialize the transmit scrambler circuit
TM_LK_SCR_CTRL[1]			**Address [0x4C808]**
TM_LK_SCR_CTRL[2]			**Address [0x4D008]**
TM_LK_SCR_CTRL[3]			**Address [0x4D808]**
TM_LK_SCR_CTRL[4]			**Address [0x4E008]**
TM_LK_SCR_CTRL[5]			**Address [0x4E808]**

8.4.1.4 TM_LK_L1_CFG[0] Register (offset = 0x4C00C)

Table 8-40. TM_LK_L1_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-5	RSVD	NOT_ACCESSIBLE	Reserved
4-0	L1_Inband_CFG	READ_WRITE	Bit 0: RESET Bit 1:RAI Bit 2: SDI Bit 3: LOS Bit 4: LOF
TM_LK_L1_CFG[1]			**Address [0x4C80C]**
TM_LK_L1_CFG[2]			**Address [0x4D00C]**
TM_LK_L1_CFG[3]			**Address [0x4D80C]**
TM_LK_L1_CFG[4]			**Address [0x4E00C]**
TM_LK_L1_CFG[5]			**Address [0x4E80C]**

8.4.1.5 TM_LK_L1_EN[0] Register (offset = 0x4C010)

Table 8-41. TM_LK_L1_EN[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD	NOT_ACCESSIBLE	Reserved
8-0	L1_Inband_EN	READ_WRITE	Bit 8: LOS_LOSERR_EN Bit 7: LOS_LOSRX_EN, Bit 6: LOF_LOFERR_EN Bit 5: LOF_LOFRX_EN Bit 4: RAI_LOSERR_EN Bit 3: RAI_LOSRX_E Bit 2: RAI_LOFERR_EN Bit 1: RAI_LOFRX_EN Bit 0: RAI_RAIRX_EN
TM_LK_L1_EN[1]			**Address [0x4C810]**
TM_LK_L1_EN[2]			**Address [0x4D010]**
TM_LK_L1_EN[3]			**Address [0x4D810]**
TM_LK_L1_EN[4]			**Address [0x4E010]**
TM_LK_L1_EN[5]			**Address [0x4E810]**

8.4.1.6 TM_LK_LOSERR[0] Register (offset = 0x4C014)

Table 8-42. TM_LK_LOSERR[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	Reserved
2-0	RM_Link_LOSERR	READ_WRITE	RM Link LOSERR Select 0 = RM Link LOSERR 0 1 = RM Link LOSERR 1 2 = RM Link LOSERR 2 3 = RM Link LOSERR 3 4 = RM Link LOSERR 4 5 = RM Link LOSERR 5
TM_LK_LOSERR[1]			**Address [0x4C814]**
TM_LK_LOSERR[2]			**Address [0x4D014]**
TM_LK_LOSERR[3]			**Address [0x4D814]**
TM_LK_LOSERR[4]			**Address [0x4E014]**
TM_LK_LOSERR[5]			**Address [0x4E814]**

8.4.1.7 TM_LK_LOFERR[0] Register (offset = 0x4C018)

Table 8-43. TM_LK_LOFERR[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	Reserved
2-0	RM_Link_LOFERR	READ_WRITE	RM Link LOFERR Select 0 = RM Link 0 1 = RM Link 1 2 = RM Link 2 3 = RM Link 3 4 = RM Link 4 5 = RM Link 5
TM_LK_LOFERR[1]			**Address [0x4C818]**
TM_LK_LOFERR[2]			**Address [0x4D018]**
TM_LK_LOFERR[3]			**Address [0x4D818]**
TM_LK_LOFERR[4]			**Address [0x4E018]**
TM_LK_LOFERR[5]			**Address [0x4E818]**

8.4.1.8 TM_LK_LOSRx[0] Register (offset = 0x4C01C)

Table 8-44. TM_LK_LOSRx[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	Reserved
2-0	RM_Link_LOSRx	READ_WRITE	RM Link LOSRx Select 0 = RM Link LOSRx 0 1 = RM Link LOSRx 1 2 = RM Link LOSRx 2 3 = RM Link LOSRx 3 4 = RM Link LOSRx 4 5 = RM Link LOSRx 5
TM_LK_LOSRx[1]			**Address [0x4C81C]**
TM_LK_LOSRx[2]			**Address [0x4D01C]**
TM_LK_LOSRx[3]			**Address [0x4D81C]**
TM_LK_LOSRx[4]			**Address [0x4E01C]**
TM_LK_LOSRx[5]			**Address [0x4E81C]**

8.4.1.9 TM_LK_LOFRx[0] Register (offset = 0x4C020)

Table 8-45. TM_LK_LOFRx[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	Reserved
2-0	RM_Link_LOFRx	READ_WRITE	RM Link LOFRx Select 0 = RM Link LOFRx 0 1 = RM Link LOFRx 1 2 = RM Link LOFRx 2 3 = RM Link LOFRx 3 4 = RM Link LOFRx 4 5 = RM Link LOFRx 5
TM_LK_LOFRx[1]			**Address [0x4C820]**
TM_LK_LOFRx[2]			**Address [0x4D020]**
TM_LK_LOFRx[3]			**Address [0x4D820]**
TM_LK_LOFRx[4]			**Address [0x4E020]**
TM_LK_LOFRx[5]			**Address [0x4E820]**

8.4.1.10 TM_LK_RAIRx[0] Register (offset = 0x4C024)

Table 8-46. TM_LK_RAIRx[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	Reserved
2-0	RM_Link_RAIRx	READ_WRITE	RM Link RAIRx Select 0 = RM Link RAIRx 0 1 = RM Link RAIRx 1 2 = RM Link RAIRx 2 3 = RM Link RAIRx 3 4 = RM Link RAIRx 4 5 = RM Link RAIRx 5
TM_LK_RAIRx[1]			**Address [0x4C824]**
TM_LK_RAIRx[2]			**Address [0x4D024]**
TM_LK_RAIRx[3]			**Address [0x4D824]**
TM_LK_RAIRx[4]			**Address [0x4E024]**
TM_LK_RAIRx[5]			**Address [0x4E824]**

8.4.1.11 TM_LK_HFN[0] Register (offset = 0x4C028)

Table 8-47. TM_LK_HFN[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved
7-0	HFN	READ	The Hyperframe number of the currently transmitting frame
TM_LK_HFN[1]			**Address [0x4C828]**
TM_LK_HFN[2]			**Address [0x4D028]**
TM_LK_HFN[3]			**Address [0x4D828]**
TM_LK_HFN[4]			**Address [0x4E028]**
TM_LK_HFN[5]			**Address [0x4E828]**

8.4.1.12 TM_LK_PTRP[0] Register (offset = 0x4C02C)

Table 8-48. TM_LK_PTRP[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved
7-0	PTR_P	READ_WRITE	CPRI Pointer P value
TM_LK_PTRP[1]			**Address [0x4C82C]**
TM_LK_PTRP[2]			**Address [0x4D02C]**
TM_LK_PTRP[3]			**Address [0x4D82C]**
TM_LK_PTRP[4]			**Address [0x4E02C]**
TM_LK_PTRP[5]			**Address [0x4E82C]**

8.4.1.13 TM_LK_STRT[0] Register (offset = 0x4C030)

Table 8-49. TM_LK_STRT[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved
7-0	STARTUP	READ_WRITE	CPRI Startup value
TM_LK_STRT[1]			**Address [0x4C830]**
TM_LK_STRT[2]			**Address [0x4D030]**
TM_LK_STRT[3]			**Address [0x4D830]**
TM_LK_STRT[4]			**Address [0x4E030]**
TM_LK_STRT[5]			**Address [0x4E830]**

8.4.1.14 TM_LK_PROT[0] Register (offset = 0x4C034)

Table 8-50. TM_LK_PROT[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved
7-0	PROT_VERS	READ_WRITE	CPRI Protocol version value
TM_LK_PROT[1]			**Address [0x4C834]**
TM_LK_PROT[2]			**Address [0x4D034]**
TM_LK_PROT[3]			**Address [0x4D834]**
TM_LK_PROT[4]			**Address [0x4E034]**
TM_LK_PROT[5]			**Address [0x4E834]**

8.4.1.15 TM_LK_STAT[0] Register (offset = 0x4C038)

Table 8-51. TM_LK_STAT[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved
6-3	FRM_STATE	READ	Indicates that status of the Frame Sync state machine 1 = FSM in OFF state 2 = FSM in IDLE state 4 = FSM in RE_SYNC state 8 = FSM in FRAME_SYNC state
2	FRM_MISALIGN	READ	CO Frame alignment mismatch from transmit FSM
1	FIFO_UNFL	READ	Tx FIFO underflow
0	TM_FAIL	READ	TM Fail condition met
TM_LK_STAT[1]			**Address [0x4C838]**
TM_LK_STAT[2]			**Address [0x4D038]**
TM_LK_STAT[3]			**Address [0x4D838]**
TM_LK_STAT[4]			**Address [0x4E038]**
TM_LK_STAT[5]			**Address [0x4E838]**

8.4.1.16 TM_FRM_MODE[0] Register (offset = 0x4C03C)

Table 8-52. TM_FRM_MODE[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved
0	FRM_MODE	READ_WRITE	Short Frame Enable Disable. 0: Disable, 1:Enable
TM_FRM_MODE[1]			**Address [0x4C83C]**
TM_FRM_MODE[2]			**Address [0x4D03C]**
TM_FRM_MODE[3]			**Address [0x4D83C]**
TM_FRM_MODE[4]			**Address [0x4E03C]**
TM_FRM_MODE[5]			**Address [0x4E83C]**

8.5 CI Registers

Table 8-53. CI Register Memory Map

Offset	Register Name	Access	Description	Section
0x58000	CI_LK_CFG[0]	R/W	CI Configuration Register (Link0)	Section 8.5.1.1
0x58800	CI_LK_CFG[1]	R/W	CI Configuration Register (Link1)	Section 8.5.1.1
0x59000	CI_LK_CFG[2]	R/W	CI Configuration Register (Link2)	Section 8.5.1.1
0x59800	CI_LK_CFG[3]	R/W	CI Configuration Register (Link3)	Section 8.5.1.1
0x5A000	CI_LK_CFG[4]	R/W	CI Configuration Register (Link4)	Section 8.5.1.1
0x5A800	CI_LK_CFG[5]	R/W	CI Configuration Register (Link5)	Section 8.5.1.1

8.5.1 Link Registers Details

8.5.1.1 CI_LK_CFG[0] Register (offset = 0x58000)

Table 8-54. CI_LK_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-5	RSVD	NOT_ACCESSIBLE	RESERVED
4-3	sample_width	READ_WRITE	The Sample Width field defines the sample size of the antenna carriers within a CPRI frame. 0 = 7 Bit Samples 1 = 8 Bit Samples 2 = 15 Bit Samples 3 = 16 Bit Samples
2	OBSAI_CPRI	READ_WRITE	The OBSAI_CPRI bit defines the mode of operation for the block. 0 = CPRI Mode 1 = OBSAI Mode
1-0	Link_rate	READ_WRITE	The link rate field defines the rate of each link that each block will provide conversion. 0 = 8X Rate 1 = 4X Rate 2 = 2X Rate 3 = 5X Rate
CI_LK_CFG[1]		**Access = READ_WRITE**	**Address [0x58800]**
CI_LK_CFG[2]		**Access = READ_WRITE**	**Address [0x59000]**
CI_LK_CFG[3]		**Access = READ_WRITE**	**Address [0x59800]**
CI_LK_CFG[4]		**Access = READ_WRITE**	**Address [0x5A000]**
CI_LK_CFG[5]		**Access = READ_WRITE**	**Address [0x5A800]**

8.6 CO Registers

Table 8-55. CO Register Memory Map

Offset	Register Name	Access	Description	Section
0x5C000	CO_LK_CFG[0]	R/W	CO Configuration Register (Link 0)	Section 8.6.1.1
0x5C800	CO_LK_CFG[1]	R/W	CO Configuration Register (Link 1)	Section 8.6.1.1
0x5D000	CO_LK_CFG[2]	R/W	CO Configuration Register (Link 2)	Section 8.6.1.1
0x5D800	CO_LK_CFG[3]	R/W	CO Configuration Register (Link 3)	Section 8.6.1.1
0x5E000	CO_LK_CFG[4]	R/W	CO Configuration Register (Link 4)	Section 8.6.1.1
0x5E800	CO_LK_CFG[5]	R/W	CO Configuration Register (Link 5)	Section 8.6.1.1

8.6.1 Link Registers Details

8.6.1.1 CO_LK_CFG[0] Register (offset = 0x5C000)

Table 8-56. CO_LK_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-5	RSVD	NOT_ACCESSIBLE	RESERVED
4-3	sample_width	READ_WRITE	The Sample Width field defines the sample size of the antenna carriers within a CPRI frame. 0 7 Bit Samples 1 8 Bit Samples 2 15 Bit Samples 3 16 Bit Samples
2	OBSAI_CPRI	READ_WRITE	The OBSAI_CPRI bit defines the mode of operation for the block. 0 CPRI Mode 1 OBSAI Mode
1-0	Link_rate	READ_WRITE	The link rate field defines the rate of each link that each block will provide conversion. 0 8X Rate 1 4X Rate 2 2X Rate 3 5X Rate
CO_LK_CFG[1]	**Access = READ_WRITE**	**Address [0x5C800]**	
CO_LK_CFG[2]	**Access = READ_WRITE**	**Address [0x5D000]**	
CO_LK_CFG[3]	**Access = READ_WRITE**	**Address [0x5D800]**	
CO_LK_CFG[4]	**Access = READ_WRITE**	**Address [0x5E000]**	
CO_LK_CFG[5]	**Access = READ_WRITE**	**Address [0x5E800]**	

8.7 RT Registers

Table 8-57. RT Register Memory Map

Offset	Register Name	Access	Description	Section
0x54000	RT_LK_CFG[0]	R/W	RT Configuration Register (Link 0)	Section 8.7.1.1
0x54004	RT_LK_DPTH[0]	R	RT FIFO DEPTH Status	Section 8.7.1.2
0x54008	RT_HDR_ERR[0]	R	RT Header Error Status	Section 8.7.1.3
0x5400C	RT_LK_STAT[0]	R	The RT Status Register contains status of the RT block	Section 8.7.1.4
0x54800	RT_LK_CFG[1]	R/W	RT Configuration Register (Link 1)	Section 8.7.1.1
0x54804	RT_LK_DPTH[1]	R	RT FIFO DEPTH Status	Section 8.7.1.2
0x54808	RT_HDR_ERR[1]	R	RT Header Error Status	Section 8.7.1.3
0x5480C	RT_LK_STAT[1]	R	The RT Status Register contains status of the RT block	Section 8.7.1.4
0x55000	RT_LK_CFG[2]	R/W	RT Configuration Register (Link 2)	Section 8.7.1.1
0x55004	RT_LK_DPTH[2]	R	RT FIFO DEPTH Status	Section 8.7.1.2
0x55008	RT_HDR_ERR[2]	R	RT Header Error Status	Section 8.7.1.3
0x5500C	RT_LK_STAT[2]	R	The RT Status Register contains status of the RT block	Section 8.7.1.4
0x55800	RT_LK_CFG[3]	R/W	RT Configuration Register (Link 3)	Section 8.7.1.1
0x55804	RT_LK_DPTH[3]	R	RT FIFO DEPTH Status	Section 8.7.1.2
0x55808	RT_HDR_ERR[3]	R	RT Header Error Status	Section 8.7.1.3
0x5580C	RT_LK_STAT[3]	R	The RT Status Register contains status of the RT block	Section 8.7.1.4
0x56000	RT_LK_CFG[4]	R/W	RT Configuration Register (Link 4)	Section 8.7.1.1
0x56004	RT_LK_DPTH[4]	R	RT FIFO DEPTH Status	Section 8.7.1.2
0x56008	RT_HDR_ERR[4]	R	RT Header Error Status	Section 8.7.1.3
0x5600C	RT_LK_STAT[4]	R	The RT Status Register contains status of the RT block	Section 8.7.1.4
0x56800	RT_LK_CFG[5]	R/W	RT Configuration Register (Link 5)	Section 8.7.1.1
0x56804	RT_LK_DPTH[5]	R	RT FIFO DEPTH Status	Section 8.7.1.2
0x56808	RT_HDR_ERR[5]	R	RT Header Error Status	Section 8.7.1.3
0x5680C	RT_LK_STAT[5]	R	The RT Status Register contains status of the RT block	Section 8.7.1.4

8.7.1 Link Registers Details

8.7.1.1 RT_LK_CFG[0] Register (offset = 0x54000)

Table 8-58. RT_LK_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-11	RSVD	NOT_ACCESSIBLE	RESERVED
10-9	RT_Config	READ_WRITE	The RT Configuration field configures the mode of operation of the RT block. 0 = Retransmit mode (Use Ingress data from RM only) 1 = Aggregate mode (WCDMA: data addition, OFDM: data insertion) 2 = Transmit mode (Use PE generated data only)
8	EM_Enable	READ_WRITE	The Empty Message Enable Bit configures the RT block to insert an empty message in the event that a message is discovered to have a code violation in the header. The configuration is valid in OBSAI mode only. 0 = Empty Message Insertion not Enabled 1 = Empty Message Insertion Enabled
7-5	CI_Select	READ_WRITE	The CI Select bits indicate the selected link for the retransmitted data path. 0 = Link 0 Selected 1 = Link 1 Selected 2 = Link 2 Selected 3 = Link 3 Selected 4 = Link 4 Selected 5 = Link 5 Selected
4-3	sample_width	READ_WRITE	The Sample Width field defines the sample size of the antenna carriers within a CPRI frame. 0 = 7 Bit Samples 1 = 8 Bit Samples 2 = 15 Bit Samples 3 = 16 Bit Samples
2	OBSAI_CPRI	READ_WRITE	The OBSAI_CPRI bit defines the mode of operation for the block. 0 = CPRI Mode 1 = OBSAI Mode
1-0	Link_rate	READ_WRITE	The link rate field defines the rate of each link that each block will provide conversion. 0 = 8X Rate 1 = 4X Rate 2 = 2X Rate 3 = 5X Rate

RT_LK_CFG[1]	**Access = READ_WRITE**	**Address [0x54800]**
RT_LK_CFG[2]	**Access = READ_WRITE**	**Address [0x55000]**
RT_LK_CFG[3]	**Access = READ_WRITE**	**Address [0x55800]**
RT_LK_CFG[4]	**Access = READ_WRITE**	**Address [0x56000]**
RT_LK_CFG[5]	**Access = READ_WRITE**	**Address [0x56800]**

8.7.1.2 RT_LK_DPTH[0] Register (offset = 0x54004)

Table 8-59. RT_LK_DPTH[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved
7-0	RT_DEPTH	READ	Describes the Current FIFO Depth
RT_LK_DPTH[1]	**Access = READ**	**Address [0x54804]**	
RT_LK_DPTH[2]	**Access = READ**	**Address [0x55004]**	
RT_LK_DPTH[3]	**Access = READ**	**Address [0x55804]**	
RT_LK_DPTH[4]	**Access = READ**	**Address [0x56004]**	
RT_LK_DPTH[5]	**Access = READ**	**Address [0x56804]**	

8.7.1.3 RT_HDR_ERR[0] Register (offset = 0x54008)

Table 8-60. RT_HDR_ERR[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved
7	HDR_ERR	READ	Indicates an Error has Occurred
6-0	DMA_CHAN	READ	Defines DMA Channel of Error

RT_HDR_ERR[1]	**Access = READ**	**Address [0x54808]**
RT_HDR_ERR[2]	**Access = READ**	**Address [0x55008]**
RT_HDR_ERR[3]	**Access = READ**	**Address [0x55808]**
RT_HDR_ERR[4]	**Access = READ**	**Address [0x56008]**
RT_HDR_ERR[5]	**Access = READ**	**Address [0x56808]**

8.7.1.4 RT_LK_STAT[0] Register (offset = 0x5400C)

Table 8-61. RT_LK_STAT[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-5	RSVD	NOT_ACCESSIBLE	Reserved
4	RT_FRM_ERR_STS	READ	Indicates a frame error of the FIFO has occurred
3	RT_OVFL_STS	READ	Indicates an Overflow of the FIFO has occurred
2	RT_UNFL_STS	READ	Indicates an Underflow of the FIFO has occurred
1	RT_EM_STS	READ	Indicates an empty message was inserted
0	RT_HDR_ERR_STS	READ	Indicates a header error has occurred
RT_LK_STAT[1]		**Access = READ**	**Address [0x5480C]**
RT_LK_STAT[2]		**Access = READ**	**Address [0x5500C]**
RT_LK_STAT[3]		**Access = READ**	**Address [0x5580C]**
RT_LK_STAT[4]		**Access = READ**	**Address [0x5600C]**
RT_LK_STAT[5]		**Access = READ**	**Address [0x5680C]**

8.8 PD Registers

Table 8-62. PD Register Memory Map

Offset	Register Name	Access	Description	Section
0x60000	pd_link_a[0]		Link-by-Link:Link Specific parameters	Section 8.8.1.1
0x60004	pd_link_b[0]		Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.8.1.2
0x60008	pd_lk_pack_cpri[0]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.8.1.3
0x60014	pd_cpri_crc[0]		Link-by-Link: PD dictates CRC to be used for each of four possible CPRI packets streams	Section 8.8.1.4
0x60018	pd_pack_map[0]		Link-by-Link: PD four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.8.1.5
0x6001C	pd_dbm[0]		Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.8.1.6
0x60020	pd_dbm_1map[0][0-3]		Link-by-Link: PD DBMF bit map1 Registers	Section 8.8.1.7
0x60030	pd_dbm_2map[0][0-2]		Link-by-Link: PD DBMF bit map2 (31-0) Register	Section 8.8.1.8
0x60080	pd_type_lut[0][0-31]		Link-by-Link: OBSAI Type Look Up Table. Allows for new OBSAI types to be defined (and reconfiguration of existing Types). Software should set these values via ROM look up	Section 8.8.1.9
0x60200	pd_cpri_id_lut[0][0-127]		Link-by-Link x128 CPRI chan per link: PD CPRI Steam LUT Register	Section 8.8.1.10
0x60400	pd_cw_lut[0][0-255]		Link-by-Link x256 CPRI CW per hyper frame: PD CPRI CW LUT Register	Section 8.8.1.11
0x60800	pd_link_a[1]		Link-by-Link:Link Specific parameters	Section 8.8.1.1
0x60804	pd_link_b[1]		Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.8.1.2
0x60808	pd_lk_pack_cpri[1]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.8.1.3
0x60814	pd_cpri_crc[1]		Link-by-Link: PD dictates CRC to be used for each of four possible CPRI packets streams	Section 8.8.1.4
0x60818	pd_pack_map[1]		Link-by-Link: PD four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.8.1.5
0x6081C	pd_dbm[1]		Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.8.1.6
0x60820	pd_dbm_1map[1][0-3]		Link-by-Link: PD DBMF bit map1 Registers	Section 8.8.1.7
0x60830	pd_dbm_2map[1][0-2]		Link-by-Link: PD DBMF bit map2 (31-0) Register	Section 8.8.1.8
0x60880	pd_type_lut[1][0-31]		Link-by-Link: OBSAI Type Look Up Table. Allows for new OBSAI types to be defined (and reconfiguration of existing Types). Software should set these values via ROM look up	Section 8.8.1.9
0x60A00	pd_cpri_id_lut[1][0-127]		Link-by-Link x128 CPRI chan per link: PD CPRI Steam LUT Register	Section 8.8.1.10
0x60C00	pd_cw_lut[1][0-255]		Link-by-Link x256 CPRI CW per hyper frame: PD CPRI CW LUT Register	Section 8.8.1.11
0x61000	pd_link_a[2]		Link-by-Link:Link Specific parameters	Section 8.8.1.1
0x61004	pd_link_b[2]		Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.8.1.2
0x61008	pd_lk_pack_cpri[2]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.8.1.3

Table 8-62. PD Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x61014	pd_cpri_crc[2]		Link-by-Link: PD dictates CRC to be used for each of four possible CPRI packets streams	Section 8.8.1.4
0x61018	pd_pack_map[2]		Link-by-Link: PD four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.8.1.5
0x6101C	pd_dbm[2]		Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.8.1.6
0x61020	pd_dbm_1map[2][0-3]		Link-by-Link: PD DBMF bit map1 Registers	Section 8.8.1.7
0x61030	pd_dbm_2map[2][0-2]		Link-by-Link: PD DBMF bit map2 (31-0) Register	Section 8.8.1.8
0x61080	pd_type_lut[2][0-31]		Link-by-Link: OBSAI Type Look Up Table. Allows for new OBSAI types to be defined (and reconfiguration of existing Types). Software should set these values via ROM look up	Section 8.8.1.9
0x61200	pd_cpri_id_lut[2][0-127]		Link-by-Link x128 CPRI chan per link: PD CPRI Steam LUT Register	Section 8.8.1.10
0x61400	pd_cw_lut[2][0-255]		Link-by-Link x256 CPRI CW per hyper frame: PD CPRI CW LUT Register	Section 8.8.1.11
0x61800	pd_link_a[3]		Link-by-Link:Link Specific parameters	Section 8.8.1.1
0x61804	pd_link_b[3]		Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.8.1.2
0x61808	pd_lk_pack_cpri[3]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.8.1.3
0x61814	pd_cpri_crc[3]		Link-by-Link: PD dictates CRC to be used for each of four possible CPRI packets streams	Section 8.8.1.4
0x61818	pd_pack_map[3]		Link-by-Link: PD four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.8.1.5
0x6181C	pd_dbm[3]		Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.8.1.6
0x61820	pd_dbm_1map[3][0-3]		Link-by-Link: PD DBMF bit map1 Registers	Section 8.8.1.7
0x61830	pd_dbm_2map[3][0-2]		Link-by-Link: PD DBMF bit map2 (31-0) Register	Section 8.8.1.8
0x61880	pd_type_lut[3][0-31]		Link-by-Link: OBSAI Type Look Up Table. Allows for new OBSAI types to be defined (and reconfiguration of existing Types). Software should set these values via ROM look up	Section 8.8.1.9
0x61A00	pd_cpri_id_lut[3][0-127]		Link-by-Link x128 CPRI chan per link: PD CPRI Steam LUT Register	Section 8.8.1.10
0x61C00	pd_cw_lut[3][0-255]		Link-by-Link x256 CPRI CW per hyper frame: PD CPRI CW LUT Register	Section 8.8.1.11
0x62000	pd_link_a[4]		Link-by-Link:Link Specific parameters	Section 8.8.1.1
0x62004	pd_link_b[4]		Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.8.1.2
0x62008	pd_lk_pack_cpri[4]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.8.1.3
0x62014	pd_cpri_crc[4]		Link-by-Link: PD dictates CRC to be used for each of four possible CPRI packets streams	Section 8.8.1.4
0x62018	pd_pack_map[4]		Link-by-Link: PD four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.8.1.5

Table 8-62. PD Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x6201C	pd_dbm[4]		Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.8.1.6
0x62020	pd_dbm_1map[4][0-3]		Link-by-Link: PD DBMF bit map1 Registers	Section 8.8.1.7
0x62030	pd_dbm_2map[4][0-2]		Link-by-Link: PD DBMF bit map2 (31-0) Register	Section 8.8.1.8
0x62080	pd_type_lut[4][0-31]		Link-by-Link: OBSAI Type Look Up Table. Allows for new OBSAI types to be defined (and reconfiguration of existing Types). Software should set these values via ROM look up	Section 8.8.1.9
0x62200	pd_cpri_id_lut[4][0-127]		Link-by-Link x128 CPRI chan per link: PD CPRI Steam LUT Register	Section 8.8.1.10
0x62400	pd_cw_lut[4][0-255]		Link-by-Link x256 CPRI CW per hyper frame: PD CPRI CW LUT Register	Section 8.8.1.11
0x62800	pd_link_a[5]		Link-by-Link:Link Specific parameters	Section 8.8.1.1
0x62804	pd_link_b[5]		Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.8.1.2
0x62808	pd_lk_pack_cpri[5]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.8.1.3
0x62814	pd_cpri_crc[5]		Link-by-Link: PD dictates CRC to be used for each of four possible CPRI packets streams	Section 8.8.1.4
0x62818	pd_pack_map[5]		Link-by-Link: PD four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.8.1.5
0x6281C	pd_dbm[5]		Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.8.1.6
0x62820	pd_dbm_1map[5][0-3]		Link-by-Link: PD DBMF bit map1 Registers	Section 8.8.1.7
0x62830	pd_dbm_2map[5][0-2]		Link-by-Link: PD DBMF bit map2 (31-0) Register	Section 8.8.1.8
0x62880	pd_type_lut[5][0-31]		Link-by-Link: OBSAI Type Look Up Table. Allows for new OBSAI types to be defined (and reconfiguration of existing Types). Software should set these values via ROM look up	Section 8.8.1.9
0x62A00	pd_cpri_id_lut[5][0-127]		Link-by-Link x128 CPRI chan per link: PD CPRI Steam LUT Register	Section 8.8.1.10
0x62C00	pd_cw_lut[5][0-255]		Link-by-Link x256 CPRI CW per hyper frame: PD CPRI CW LUT Register	Section 8.8.1.11
0x63000	pd_global		Fields that are global for the entire PD	Section 8.8.1.12
0x63004	db_pd_global_en_set		Set Global Enable for PD	Section 8.8.1.13
0x63008	db_pd_global_en_clr		Clear Global Enable for PD	Section 8.8.1.14
0x6300c	pd_global_en_sts		Read Only status of global enable state. Even if this register is OFF, PD may still be closing out packets.	Section 8.8.1.15
0x63010	pd_dma		Controls DMA functionality that is common among all links	Section 8.8.1.16
0x63014	pd_radt_tc		OBSAI timing control, RadT usage (used for OBSAI LTE and WiMax	Section 8.8.1.17
0x63020	pd_chan_sts[4]		Gives current On/Off Status of every available stream. One bit per channel. Required because channels only turn on/off on radio frame or packet frame boundaries so the chan_en alone does not give channel status (Each of 4 registers represents 32 of 128 channels)	Section 8.8.1.18
0x63030	pd_pkt_sts[4]		Gives current In/Out Packet Status of every available stream. One bit per channel. Helps debug if PD is not shutting down	Section 8.8.1.19

Table 8-62. PD Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x63040	pd_frm_tc[6]		For framing state machine, supplies terminal counts of some of the state counters and some start counts. Six different versions of this register are used to support LTE variations, 3 bandwidths and 2 symbol lengths simultaneously	Section 8.8.1.20
0x63400	pd_route[128]		x128 DMA chan: Information used to decode OBSAI header information for routing into DMA channels.	Section 8.8.1.21
0x63600	pd_dmachan[128]		x128 DMA chan: PD Channel Control Fields Register	Section 8.8.1.22
0x64000	pd_dmachan_a[128]		x128 DMA chan: PD Channel Control Fields Register 0	Section 8.8.1.23
0x64200	pd_dmachan_b[128]		x128 DMA chan: PD Channel Control Fields Register 1	Section 8.8.1.24
0x64400	pd_dmachan_c[128]		x128 DMA chan: PD Channel Control Fields Register 2	Section 8.8.1.25
0x64600	pd_dmachan_d[128]		x128 DMA chan: PD Channel Control Fields Register 3	Section 8.8.1.26
0x64800	pd_dmachan_e[128]		x128 DMA chan: PD Channel Control Fields Register 4	Section 8.8.1.27
0x64A00	pd_dmachan_f[128]		x128 DMA chan: PD Channel Control Fields Register 5	Section 8.8.1.28
0x64C00	pd_frm_msg_tc[128]		x128 LUT Depth: PD Frame Message Count Register	Section 8.8.1.29

8.8.1 Link Registers Details

8.8.1.1 pd_link_a[0] Register (offset = 0x60000)

Table 8-63. pd_link_a[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-24	crc8_seed	READ_WRITE	CRC8: programmable starting seed value (CRC16 and CRC32 seed is fixed)
23-16	crc8_poly	READ_WRITE	CRC8: programmable polynomial for CRC8 (CRC16 and CRC32 polynomials are fixed)
15-12	RSVD2	NOT_ACCESSIBLE	RESERVED
11-8	ethnet_strip	READ_WRITE	CPRI: bit3-0 enables stripping Ethernet headers for chan3-0 OBSAI: unused
7-5	RSVD1	NOT_ACCESSIBLE	RESERVED
4	obsai_cpri	READ_WRITE	OBSAI or CPRI mode of operation. 0 = CPRI 1 = OBSAI
3-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	link_en	READ_WRITE	Link Enable/Disable
pd_link_a[1]			**Address [0x60800]**
pd_link_a[2]			**Address [0x61000]**
pd_link_a[3]			**Address [0x61800]**
pd_link_a[4]			**Address [0x62000]**
pd_link_a[5]			**Address [0x62800]**

8.8.1.2 pd_link_b[0] Register (offset = 0x60004)

Table 8-64. pd_link_b[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-30	pkt_delim_ch3	READ_WRITE	CPRI Control Word & Packet encoding option
29-28	pkt_delim_ch2	READ_WRITE	CPRI Control Word & Packet encoding option
27-26	pkt_delim_ch1	READ_WRITE	CPRI Control Word & Packet encoding option
25-24	pkt_delim_ch0	READ_WRITE	CPRI Control Word & Packet encoding option 0 = User defined data is not extracted from CPRI Control Words 1 = CW user data is delimitted using 4B/5B encoding where SOP and EOP are defined 2 = CW user data is delimitted by a user defined Null Delimiter (defined in other MMR field) 3 = Packet Boundaries are inferred to be on hyper-frame boundaries
23-21	RSVD1	NOT_ACCESSIBLE	RESERVED
20-12	cpri_nulldelm	READ_WRITE	CPRI Control Word Null Delimiter. Used instead of 4B/5B for packet delimitation. Only used when enabled by pkt_delim_ch fields
11-4	RSVD	NOT_ACCESSIBLE	RESERVED
3-2	cpri_axc_pack	READ_WRITE	CRPI: identifies the bit precision of the IQ data. Used to unpacking packet data passed over CPRI AxC slots 0 = 7 Bit Samples:AxC data is 7bit I and 7bit Q (byte packing needed) 1 = 8 Bit Samples:AxC data is 8bit I and 8bit Q (byte aligned) 2 = 15 Bit Samples:AxC data is 15bit I and 15bit Q (byte packing needed) 3 = 16 Bit Samples:AxC data is 16bit I and 16bit Q (byte aligned)
1-0	cpri_lk_rate	READ_WRITE	CPRI link rate (na for OBSAI). 0 = 8X_RATE 1 = 4x_RATE 2 = 2X_RATE 3 = 5x_RATE
pd_link_b[1]			**Address [0x60804]**
pd_link_b[2]			**Address [0x61004]**
pd_link_b[3]			**Address [0x61804]**
pd_link_b[4]			**Address [0x62004]**
pd_link_b[5]			**Address [0x62804]**

8.8.1.3 pd_lk_pack_cpri[0] Register (offset = 0x60008)

Table 8-65. pd_lk_pack_cpri[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	RESERVED
7-4	post_enc_bitswap	READ_WRITE	Swaps bits in bytes after 4B,5B,Null encoders. Bit0 for control channel 0 ~ bit3 for control channel 3. this makes control, packet data order like little endian
3-0	pre_enc_bitswap	READ_WRITE	Swaps bits in bytes before 4B,5B,Null encoders. Bit0 for control channel 0 ~ bit3 for control channel 3. this makes control, packet data order like little endian
pd_lk_pack_cpri[1]		**Access = READ_WRITE**	**Address [0x60808]**
pd_lk_pack_cpri[2]		**Access = READ_WRITE**	**Address [0x61008]**
pd_lk_pack_cpri[3]		**Access = READ_WRITE**	**Address [0x61808]**
pd_lk_pack_cpri[4]		**Access = READ_WRITE**	**Address [0x62008]**
pd_lk_pack_cpri[5]		**Access = READ_WRITE**	**Address [0x62808]**

8.8.1.4 pd_cpri_crc[0] Register (offset = 0x60014)

Table 8-66. pd_cpri_crc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-15	RSVD3	NOT_ACCESSIBLE	RESERVED
14	crc3_en	READ_WRITE	(same) CPRI Packt Stream 3
13-12	crc3_type	READ_WRITE	(same) CPRI Packt Stream 3
11	RSVD2	NOT_ACCESSIBLE	RESERVED
10	crc2_en	READ_WRITE	(same) CPRI Packt Stream 2
9-8	crc2_type	READ_WRITE	(same) CPRI Packt Stream 2
7	RSVD1	NOT_ACCESSIBLE	RESERVED
6	crc1_en	READ_WRITE	(same) CPRI Packt Stream 1
5-4	crc1_type	READ_WRITE	(same) CPRI Packt Stream 1
3	RSVD0	NOT_ACCESSIBLE	RESERVED
2	crc0_en	READ_WRITE	CRC: enable CRC check on type-by-type basis.
1-0	crc0_type	READ_WRITE	CRC: length of CRC for Packet Stream 0 0 = 32BIT_CRC 1 = 16BIT_CRC 2 = 8BIT_CRC 3 = Reserved
pd_cpri_crc[1]			**Address [0x60814]**
pd_cpri_crc[2]			**Address [0x61014]**
pd_cpri_crc[3]			**Address [0x61814]**
pd_cpri_crc[4]			**Address [0x62014]**
pd_cpri_crc[5]			**Address [0x62814]**

8.8.1.5 pd_pack_map[0] Register (offset = 0x60018)

Table 8-67. pd_pack_map[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	pack3_en	READ_WRITE	Enables usage of pack3 circuit. When disabled, data is dropped on the floor without capture to a DMA channel
30-24	pack3_dma_ch	READ_WRITE	When enabled, indicates which DMA channel should be associated with this stream (Set 0 ~ 127)
23	pack2_en	READ_WRITE	same... pack2
22-16	pack2_dma_ch	READ_WRITE	same... pack2
15	pack1_en	READ_WRITE	same... pack1
14-8	pack1_dma_ch	READ_WRITE	same... pack1
7	pack0_en	READ_WRITE	same... pack0
6-0	pack0_dma_ch	READ_WRITE	same... pack0
pd_pack_map[1]			**Address [0x60818]**
pd_pack_map[2]			**Address [0x61018]**
pd_pack_map[3]			**Address [0x61818]**
pd_pack_map[4]			**Address [0x62018]**
pd_pack_map[5]			**Address [0x62818]**

8.8.1.6 pd_dbm[0] Register (offset = 0x6001C)

Table 8-68. pd_dbm[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD1	NOT_ACCESSIBLE	RESERVED
30-24	dbm_2size	READ_WRITE	DBMF number of bits of bit map2 to use. Set n value
23	RSVD	NOT_ACCESSIBLE	RESERVED
22-16	dbm_1size	READ_WRITE	DBMF number of bits of bit map1 to use. Set n-1 value
15-11	dbm_1mult	READ_WRITE	DBMF repetitions of map1. set N-1 value
10-7	dbm_xbubble	READ_WRITE	DBMF bubble count where 0:indicates 1 bubble of 1 AxC sample (4bytes). OBSAI DBMF assumes 4 AxC samples as a bubble length. A burst of bubbles is inserted only when dictated to do so by the bit maps. This field gives great flexibility in rate matching
6-0	dbm_x	READ_WRITE	CPRI Dual Bit Map FSM (DBMF) channel or sample count, sets the max number of AxC supported in a given Link. that is, for CPRI 4x LTE15MHz this must be set to 7b'0000001 indicating 2 AxC. For TD-SCDMA, X count could be different to the DB or AxC channel number.
pd_dbm[1]			**Address [0x6081C]**
pd_dbm[2]			**Address [0x6101C]**
pd_dbm[3]			**Address [0x6181C]**
pd_dbm[4]			**Address [0x6201C]**
pd_dbm[5]			**Address [0x6281C]**

8.8.1.7 pd_dbm_1map[0][0-3] Register (offset = 0x60020)

Table 8-69. pd_dbm_1map[0][0-3] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	dbm_1map	READ_WRITE	bit-by-bit map [99-0]. pd_dbm_1map[0].dbm_1map[0] == first 32 bit from 0 of 100 bits
pd_dbm_1map[1][0-3]			**Address [0x60820 - 0x6082F]**
pd_dbm_1map[2][0-3]			**Address [0x61020 - 0x6102F]**
pd_dbm_1map[3][0-3]			**Address [0x61820 - 0x6182F]**
pd_dbm_1map[4][0-3]			**Address [0x62020 - 0x6202F]**
pd_dbm_1map[5][0-3]			**Address [0x62820 - 0x6282F]**

8.8.1.8 pd_dbm_2map[0][0-2] Register (offset = 0x60030)

Table 8-70. pd_dbm_2map[0][0-2] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	dbm_2map	READ_WRITE	bit-by-bit map [67-0]. pd_dbm_2map[0].dbm_2map[0] == first 32 bit from 0 of 68 bits
pd_dbm_2map[1][0-2]			**Address [0x60830 - 0x6083F]**
pd_dbm_2map[2][0-2]			**Address [0x61030 - 0x6103F]**
pd_dbm_2map[3][0-2]			**Address [0x61830 - 0x6183F]**
pd_dbm_2map[4][0-2]			**Address [0x62030 - 0x6203F]**
pd_dbm_2map[5][0-2]			**Address [0x62830 - 0x6283F]**

8.8.1.9 pd_type_lut[0][0-31] Register (offset = 0x60080)

Table 8-71. pd_type_lut[0][0-31] Register Field Descriptions

Bits	Field Name	Type	Description
31-17	RSVD4	NOT_ACCESSIBLE	RESERVED
16	crc_hdr	READ_WRITE	CRC calculation for OBSAI header 1 = CRC16 is calculated over 3 byte OBSAI header and 14 bytes of 16 byte payload. Only valid for 19 byte OBSAI messages utilizing CRC16 0 = Do not perform CRC over OBSAI Header (program to OFF for CPRI and for most OBSAI types)
15-12	RSVD3	NOT_ACCESSIBLE	RESERVED
11	enet_strip	READ_WRITE	0x0: No Strip 0x1: Strip off the first eight bytes of each packet. Purpose is to strip the ethernet preamble and SOP. PD blindly strips without checking content of eight bytes. CRC is not calculated over stripped bytes.
10-9	RSVD2	NOT_ACCESSIBLE	RESERVED
8	obsai_pkt_en	READ_WRITE	Used to determine if PD should search for Radio Frame Boundary or if alignment is a don't care. determines if channel is packet type channel or axc type channel. AXC channels are expected to stream where packet channels are on-demand. AXC channels use OBSAI Timestamp and AXC Offset while packet traffic does not. There is some redundancy between this fields and TS fields 0 = OBSAI_AXC 1 = OBSAI_PKT
7	RSVD1	NOT_ACCESSIBLE	RESERVED
6	crc_en	READ_WRITE	CRC: enable CRC check on type-by-type basis.
5-4	crc_type	READ_WRITE	CRC: length of CRC. 0 = 32BIT_CRC 1 = 16BIT_CRC 2 = 8BIT_CRC 3 = Reserved
3	RSVD	NOT_ACCESSIBLE	RESERVED
2-0	ts_format	READ_WRITE	TS: OBSAI timestamp check (n/a CPRI) 0 = No timestamp check, might be useful for some future radio standard or debug 1 = normal timestamp format 2 = GSM OBSAI Timestamp, UL timestamp format msb=1 first four msg. DL timestamp format msb=1 first msg 3 = Generic Packet (SOP=10, MOP=00, EOP=11) 4 = Ethernet Type where last TS indicates number of valid bytes in last xfer 5 = TS checked by PD_Route, one message packet with inferred SOP and EOP in same OBSAI message
pd_type_lut[1][0-31]			**Address [0x60880 - 0x608FF]**
pd_type_lut[2][0-31]			**Address [0x61080 - 0x610FF]**
pd_type_lut[3][0-31]			**Address [0x61880 - 0x618FF]**
pd_type_lut[4][0-31]			**Address [0x62080 - 0x620FF]**
pd_type_lut[5][0-31]			**Address [0x62880 - 0x628FF]**

8.8.1.10 pd_cpri_id_lut[0][0-127] Register (offset = 0x60200)

Table 8-72. pd_cpri_id_lut[0][0-127] Register Field Descriptions

Bits	Field Name	Type	Description
31-15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-12	cpri_8wd_ofset	READ_WRITE	Word level AxC offset. Used in the front end of PD to align word data into QWords. bit [1-0] are offset into a QWord. bit[2] give RSA double QWord alignment.
11-9	RSVD	NOT_ACCESSIBLE	RESERVED
8	cpri_pkt_en	READ_WRITE	DBMF CPRI Stream LUT: dictates the cpri payload is to be used as AxC (normal) or Packet traffic. (note: the concept of packet traffic in CPRI payload is a feature extension of CPRI) 0 = CPRI_AXC 1 = CPRI_PKT
7	cpri_x_en	READ_WRITE	DBMF CPRI Stream LUT: enable-disable of channel, for each value of DBM X count.
6-0	cpri_dmachan	READ_WRITE	DBMF CPRI Stream LUT: AxC:used to map DBM X (max X is 128) count to DB DMA channel. PKT:2 lsb indicates 0-3 PKT packing circuit (assumed 4B/5B encoding)
pd_cpri_id_lut[1][0-127]			**Address [0x60A00 - 0x60BFF]**
pd_cpri_id_lut[2][0-127]			**Address [0x61200 - 0x613FF]**
pd_cpri_id_lut[3][0-127]			**Address [0x61A00 - 0x61BFF]**
pd_cpri_id_lut[4][0-127]			**Address [0x62200 - 0x623FF]**
pd_cpri_id_lut[5][0-127]			**Address [0x62A00 - 0x62BFF]**

8.8.1.11 pd_cw_lut[0][0-255] Register (offset = 0x60400)

Table 8-73. pd_cw_lut[0][0-255] Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	RESERVED
3	hypfm_eop	READ_WRITE	All possible CPRI CW per hyperframe. Identifies last CW per HyperFrame for this channel. We have max 4 different streams for CW, so it can not be guaranteed which control slot is the last one for the specific stream. (It is only used when HyperFrame delimeter option is chosen from pkt_delim_ch[4] register)
2	cw_en	READ_WRITE	All possible CPRI CW per hyperframe. Dictates whether the control word should be captured at all
1-0	cw_chan	READ_WRITE	All possible CPRI CW per hyperframe are mapped to one of four CPRI CW staging areas. This allow CPRI CW to be split into 4 different streams. (set 0 ~ 3)
pd_cw_lut[1][0-255]			**Address [0x60C00 - 0x60FFF]**
pd_cw_lut[2][0-255]			**Address [0x61400 - 0x617FF]**
pd_cw_lut[3][0-255]			**Address [0x61C00 - 0x61FFF]**
pd_cw_lut[4][0-255]			**Address [0x62400 - 0x627FF]**
pd_cw_lut[5][0-255]			**Address [0x62C00 - 0x62FFF]**

8.8.1.12 pd_global Register (offset = 063000)

Table 8-74. pd_global Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD1	NOT_ACCESSIBLE	RESERVED
27-16	axcoffset_win	READ_WRITE	Reception timing window width parameter. Indicates the window size for searching the radio frame boundary identified by TS=0 The unit of this field is sys clock and not used for CPRI
15-2	RSVD	NOT_ACCESSIBLE	RESERVED
1	dio_cppi	READ_WRITE	For streams that are AxC, use DB as a circular RAM which will feed DIO DMA. (WCDMA use DIO) otherwise, DB is used as a FIFO for AxC traffic and will feed PKTDMA. (For packet streams, this fields has no impact) 0 = Multicore Navigator 1 = DIO
0	shrt_frm_mode	READ_WRITE	Short Frame mode, used for hardware test only. Shortens the OBSAI and CPRI frames for purposes of reducing simulation time. 0 = SHRT_FRM_OFF 1 = SHRT_FRM_ON

8.8.1.13 pd_global_en_set Register (0x63004)

Table 8-75. pd_global_en_set Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register that sets (enables) global enable

8.8.1.14 pd_global_en_clr Register (offset = 0x63008)

Table 8-76. pd_global_en_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register that clears (disables) global enable

8.8.1.15 pd_global_en_sts Register (offset = 0x6300C)

Table 8-77. pd_global_en_sts Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	enable	READ	0x1: PD_ON 0x0:PD_OFF

8.8.1.16 pd_dma Register (offset = 0x63010)

Table 8-78. pd_dma Register Field Descriptions

Bits	Field Name	Type	Description
31-14	RSVD	NOT_ACCESSIBLE	RESERVED
13	wdog_ee_ctrl	READ_WRITE	Report every missed WDog fail, or report only fails of missing EOP 0 = REPORT_ALL 1 = REPORT_EOP
12	wdog_eop_add	READ_WRITE	Controls circuit to simply report or to report and add an EOP if the wdog fails on an expected EOP 0 = Watch Dog mechanism used for EE reporting only 1 = Watch Dog mechanism adds an EOP in the event that the expected EOP timed out
11-0	ts_wdog_cnt	READ_WRITE	OBSAI Watch Dog Count Duration. Counts in 307.2 MHz SYS Clock. Terminal Count must be reach approximately 3 times before Timer fails

8.8.1.17 pd_radt_tc Register (offset = 0x63014)

Table 8-79. pd_radt_tc Register Field Descriptions

Bits	Field Name	Type	Description
31-25	RSVD	NOT_ACCESSIBLE	RESERVED
24-0	radt_tc	READ_WRITE	Max expected value for the RadT frame timer value (AT Clock TC for a whole Rad frame). Used for creating an executed timing window (for which TS=0 marks a Radio Frame Boundary)

8.8.1.18 pd_chan_sts[4] Register (offset = 0x63020)

Table 8-80. pd_chan_sts[4] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	chan_on	READ	0x1: CHAN_ON 0x0:CHAN_OFF

8.8.1.19 pd_pkt_sts[4] Register (offset = 0x63030)

Table 8-81. pd_pkt_sts[4] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	chan_pkt	READ	Channel is currently in the middle of a packet. Channel will not turn off even if disabled. 0 = OUT_PKT 1 = IN_PKT

8.8.1.20 pd_frm_tc[6] Register (offset = 0x63040)

Table 8-82. pd_frm_tc[6] Register Field Descriptions

Bits	Field Name	Type	Description
31-24	RSVD	NOT_ACCESSIBLE	RESERVED
23-16	frm_sym_tc	READ_WRITE	Radio framing counter. Symbol terminal count.
15-8	frm_index_sc	READ_WRITE	Radio framing counter. Index Start count.
7-0	frm_index_tc	READ_WRITE	Radio framing counter. Index Terminal count

8.8.1.21 pd_route[128] Register (offset = 0x63400)

Table 8-83. pd_route[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-30	RSVD1	NOT_ACCESSIBLE	RESERVED
29-28	route_mask	READ_WRITE	Controls how many OBSAI timestamp bits to use in the reception routing. 0 = None. Do not use the TS field for routing 1 = 4 lsb bits: Use TS(3-0) 2 = all Use TS(5-0) 3 = Reserved
27	RSVD0	NOT_ACCESSIBLE	RESERVED
26-24	route_lk	READ_WRITE	Link number for which to apply this routing rule.
23-11	route_adr	READ_WRITE	OBSAI address. 13 bit unique address that corresponded to specific channel in DSP chain
10-6	route_type	READ_WRITE	OBSAI type.
5-0	route_ts	READ_WRITE	Reception Routing: OBSAI timestamp. Used to extend addressing by using the TS field. (Only known use is for OBSAI Generic Packet type and Control Packet type) The number of bits of TS to compare is controlled via the ctrl field.

8.8.1.22 pd_dmachan[128] Register (offset = 0x63600)

Table 8-84. pd_dmachan[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD1	NOT_ACCESSIBLE	RESERVED
8	data_format	READ_WRITE	Data Format 0 = Data is byte data coming over PHY. PD packs first payload byte received into byte0 or DB QWord 1 = Data is 1/2 word data. First 16bit received is loaded into byte 1 and byte 2 of DB QWord
7-1	RSVD0	NOT_ACCESSIBLE	RESERVED
0	chan_en	READ_WRITE	Channel On/Off. Schedules a channel to be turned on or off. For AxC traffic, Channel will turn On/Off on next Symbol Boundary. For Packet Traffic, Channel will turn On/Off once channel is in EOP state, completing any packet that may currently be open 0 = CHANNEL_OFF 1 = CHANNEL_ON

8.8.1.23 pd_dmachan_a[128] Register (offset = 0x64000)

Table 8-85. pd_dmachan_a[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-25	RSVD	NOT_ACCESSIBLE	RESERVED
24-0	axc_offset	READ_WRITE	OBSAI: Antenna Carrier offset programmed in 307.2MHz clocks and relative to the RadT frame boundary. CPRI: Antenna Carrier Offset programmed in the number of four samples (QW), relative to the link (PHY) frame boundary.

8.8.1.24 pd_dmachan_b[128] Register (offset = 0x64200)

Table 8-86. pd_dmachan_b[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	tdd_en	READ_WRITE	PD TDD symbol 0 ~ 15, enables or disables DMA of some symbols (Multicore Navigator packets), Program as 0xffff for most applications Bit mapped, bit0 --> sym0 of framing FSM 1 = symbol dma enabled (each bit) 0 = symbol dma disabled (each bit)
15-12	dio_offset	READ_WRITE	Offset for DIO DMA mode. Used to align DMA for AxC with different AxC offsets. Zero means zero offset and when it is increased one, the actual offset will be increased by 4 or 8 WCDMA chips (DL, UL)
11-8	RSVD1	NOT_ACCESSIBLE	RESERVED
7-5	frm_grp	READ_WRITE	OBSAI framing counter: Which of 6 framing counter terminal counts should be used for each dma channel. (codes 3'b110 and 3'b111 are reserved)
4	gsm_ul	READ_WRITE	Data Format 0 = Not GSM UL 1 = GSM UL, has special OBSAI Timestamp implications. UL timestamp format msb=1 first four msg
3-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ts_wdog_en	READ_WRITE	TS: enable OBSAI timestamp watch dog timer (only known use is GSM BB hopping). (Set to 0 for CPRI)

8.8.1.25 pd_dmachan_c[128] Register (offset = 0x64400)

Table 8-87. pd_dmachan_c[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	tdd_en	READ_WRITE	PD TDD symbol 16 ~ 47, enable or disables DMA of some symbols (Multicore Navigator packets), Program as 0xffff for most applications Bit mapped, bit0 --> sym 16 of framing FSM 1 = symbol dma enabled (each bit) 0 = symbol dma disabled (each bit)

8.8.1.26 pd_dmachan_d[128] Register (offset = 0x64600)

Table 8-88. pd_dmachan_d[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	tdd_en	READ_WRITE	PD TDD symbol 48 ~ 79, enable or disables DMA of some symbols (Multicore Navigator packets), Program as 0xffff for most applications Bit mapped, bit0 --> sym 48 of framing FSM 1 = symbol dma enabled (each bit) 0 = symbol dma disabled (each bit)

8.8.1.27 pd_dmachan_e[128] Register (offset = 0x64800)

Table 8-89. pd_dmachan_e[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	tdd_en	READ_WRITE	PD TDD symbol 80 ~ 111, enable or disables DMA of some symbols (Multicore Navigator packets), Program as 0xffff for most applications Bit mapped, bit0 --> sym 80 of framing FSM 1 = symbol dma enabled (each bit) 0 = symbol dma disabled (each bit)

8.8.1.28 pd_dmachan_f[128] Register (offset = 0x64A00)

Table 8-90. pd_dmachan_f[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	tdd_en	READ_WRITE	PD TDD symbol 112 ~ 143, enable or disables DMA of some symbols (Multicore Navigator packets), Program as 0xffff for most applications Bit mapped, bit0 --> sym 112 of framing FSM 1 = symbol dma enabled (each bit) 0 = symbol dma disabled (each bit)

8.8.1.29 pd_frm_msg_tc[128] Register (offset = 64C00)

Table 8-91. pd_frm_msg_tc[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	RSVD	NOT_ACCESSIBLE	RESERVED
15-0	frm_msg_tc	READ_WRITE	Framing message terminal count table OBSAI: number of 16 byte message terminal count. CPRI: number of 16 byte (4x 4bytes) sample

8.9 PE Registers

Table 8-92. PE Register Memory Map

Offset	Register Name	Access	Description	Section
0x68000	pe_link[0]	R/W	PE Link Register	Section 8.9.1.1
0x68004	pe_crc[0]		CRC configuration	Section 8.9.1.2
0x68008	pe_cpri_dbm[0]	R/W	Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.9.1.3
0x6800C	pe_cpri_bitswap[0]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.9.1.4
0x68010	pe_cpridbm_1map[0][0-3]		Link-by-Link: PE DBMF bit map1 (127-0) Register	Section 8.9.1.5
0x68020	pe_cpridbm_2map[0][0-2]		Link-by-Link: PE DBMF bit map2 (95-0) Register	Section 8.9.1.6
0x6802C	pe_cpri0[0]	R/W	Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.9.1.7
0x68030	pe_cpri1[0]	R/W	Link-by-Link: PE four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.9.1.8
0x68400	pe_cpri_cw_lut[0][0-255]		Link-by-Link x256 CPRI CW per hyper frame: pe CPRI CW LUT Register	Section 8.9.1.9
0x68800	pe_link[1]	R/W	PE Link Register	Section 8.9.1.1
0x68804	pe_crc[1]		CRC configuration	Section 8.9.1.2
0x68808	pe_cpri_dbm[1]	R/W	Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.9.1.3
0x6880C	pe_cpri_bitswap[1]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.9.1.4
0x68810	pe_cpridbm_1map[1][0-3]		Link-by-Link: PE DBMF bit map1 (127-0) Register	Section 8.9.1.5
0x68820	pe_cpridbm_2map[1][0-2]		Link-by-Link: PE DBMF bit map2 (95-0) Register	Section 8.9.1.6
0x6882C	pe_cpri0[1]	R/W	Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.9.1.7
0x68830	pe_cpri1[1]	R/W	Link-by-Link: PE four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.9.1.8
0x68C00	pe_cpri_cw_lut[1][0-255]		Link-by-Link x256 CPRI CW per hyper frame: pe CPRI CW LUT Register	Section 8.9.1.9
0x69000	pe_link[2]	R/W	PE Link Register	Section 8.9.1.1
0x69004	pe_crc[2]		CRC configuration	Section 8.9.1.2
0x69008	pe_cpri_dbm[2]	R/W	Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.9.1.3
0x6900C	pe_cpri_bitswap[2]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.9.1.4
0x69010	pe_cpridbm_1map[2][0-3]		Link-by-Link: PE DBMF bit map1 (127-0) Register	Section 8.9.1.5
0x69020	pe_cpridbm_2map[2][0-2]		Link-by-Link: PE DBMF bit map2 (95-0) Register	Section 8.9.1.6
0x6902C	pe_cpri0[2]	R/W	Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.9.1.7
0x69030	pe_cpri1[2]	R/W	Link-by-Link: PE four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.9.1.8

Table 8-92. PE Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x69400	pe_cpri_cw_lut[2][0-255]		Link-by-Link x256 CPRI CW per hyper frame: pe CPRI CW LUT Register	Section 8.9.1.9
0x69800	pe_link[3]	R/W	PE Link Register	Section 8.9.1.1
0x69804	pe_crc[3]		CRC configuration	Section 8.9.1.2
0x69808	pe_cpri_dbm[3]	R/W	Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.9.1.3
0x6980C	pe_cpri_bitswap[3]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.9.1.4
0x69810	pe_cpridbm_1map[3][0-3]		Link-by-Link: PE DBMF bit map1 (127-0) Register	Section 8.9.1.5
0x69820	pe_cpridbm_2map[3][0-2]		Link-by-Link: PE DBMF bit map2 (95-0) Register	Section 8.9.1.6
0x6982C	pe_cpri0[3]	R/W	Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.9.1.7
0x69830	pe_cpri1[3]	R/W	Link-by-Link: PE four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.9.1.8
0x69C00	pe_cpri_cw_lut[3][0-255]		Link-by-Link x256 CPRI CW per hyper frame: pe CPRI CW LUT Register	Section 8.9.1.9
0x6A000	pe_link[4]	R/W	PE Link Register	Section 8.9.1.1
0x6A004	pe_crc[4]		CRC configuration	Section 8.9.1.2
0x6A008	pe_cpri_dbm[4]	R/W	Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.9.1.3
0x6A00C	pe_cpri_bitswap[4]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.9.1.4
0x6A010	pe_cpridbm_1map[4][0-3]		Link-by-Link: PE DBMF bit map1 (127-0) Register	Section 8.9.1.5
0x6A020	pe_cpridbm_2map[4][0-2]		Link-by-Link: PE DBMF bit map2 (95-0) Register	Section 8.9.1.6
0x6A02C	pe_cpri0[4]	R/W	Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.9.1.7
0x6A030	pe_cpri1[4]	R/W	Link-by-Link: PE four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.9.1.8
0x6A400	pe_cw_lut[4][0-255]		Link-by-Link x256 CPRI CW per hyper frame: pe CPRI CW LUT Register	Section 8.9.1.9
0x6A800	pe_link[5]	R/W	PE Link Register	Section 8.9.1.1
0x6A804	pe_crc[5]		CRC configuration	Section 8.9.1.2
0x6A808	pe_cpri_dbm[5]	R/W	Link-by-Link: Parameters for configuring the CPRI Dual Bit Map FSM. CPRI DBM is used to rate matching and unpacking radio standards with non-WCDMA like sampling frequency	Section 8.9.1.3
0x6A80C	pe_cpri_bitswap[5]	R/W	Bit swap controls for each control channel before or after 4B,5B,Null decoding	Section 8.9.1.4
0x6A810	pe_cpridbm_1map[5][0-3]		Link-by-Link: PE DBMF bit map1 (127-0) Register	Section 8.9.1.5
0x6A820	pe_cpridbm_2map[5][0-2]		Link-by-Link: PE DBMF bit map2 (95-0) Register	Section 8.9.1.6
0x6A82C	pe_cpri0[5]	R/W	Link-by-Link:Link Specific parameters relating to CPRI usage only	Section 8.9.1.7
0x6A830	pe_cpri1[5]	R/W	Link-by-Link: PE four custom packet packing circuits for CPRI CW and AxC usage as packet traffic. This register maps which DMA channel should be used for each of the four custom pack circuits	Section 8.9.1.8

Table 8-92. PE Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x6AC00	pe_cpri_cw_lut[5][0-255]		Link-by-Link x256 CPRI CW per hyper frame: pe CPRI CW LUT Register	Section 8.9.1.9
0x6c000	pe_global		PE Global Register	Section 8.9.1.10
0x6c004	pe_global_en_set		Set Global Enable for PE	Section 8.9.1.11
0x6c008	pe_global_en_clr		Clear Global Enable for PE	Section 8.9.1.12
0x6c00c	pe_global_en_sts		Read Only status of global enable state. Even if this register is OFF, PE may still be closing out packets.	Section 8.9.1.13
0x6c010	pe_chan_sts[4]		Gives current On/Off Status of every available stream. One bit per channel. Required because channels only turn on/off on radio frame or packet frame boundaries so the chan_en alone does not give channel status (Each of 4 registers represents 32 of 128 channels)	Section 8.9.1.14
0x6c020	pe_pkt_sts[4]		Gives current On/Off Status of every available stream. One bit per channel. Required because channels only turn on/off on radio frame or packet frame boundaries so the chan_en alone does not give channel status (Each of 4 registers represents 32 of 128 channels)	Section 8.9.1.15
0x6c080	pe_frm_tc[6]		PE Frame Count Register. There are six sets of these count value in order to support 6 simultaneous LTE configurations (Use only 1 for other standards)	Section 8.9.1.16
0x6c200	pe_dmachan_en[128]		PE DMA Channel Enable Register	Section 8.9.1.17
0x6c400	pe_dma0chan[128]		PE DMA Channel 0 Register	Section 8.9.1.18
0x6c600	pe_in_fifo[128]		PE DMA Channel 1 Register	Section 8.9.1.19
0x6c800	pe_axc_offset[128]		PE DMA Channel 1 Register	Section 8.9.1.20
0x6ca00	pe_frm_msg_tc[128]		PE AxC Framing Counter Register	Section 8.9.1.21
0x6cc00	pe_modtxrule[64]		PE Modulo Terminal Count Register	Section 8.9.1.22
0x6ce00	pe_obsai_hdr[0-127]		PE DMA Channel which controls the creation of OBSAI headers	Section 8.9.1.23
0x70000	pe_obsai_dbm[64]		PE OBSAI DBMF Register (CPRI is handled in different MMRs)	Section 8.9.1.24
0x70800	pe_dbm_map[512]		PE DBMF Bit Map 1 and 2 Register, four locations for each of 64 different rules (3 address LSBs address 0-7 entries per each of 64 rules) (MSB address==0 bit map1 MSB addr==1 bit map2. 000:map1(31-0) 001:map1(63-32) 010:map1(95-64) 01l:map1(127-96) 100:map2(31-0) 101:map2(63-32) 110:map2(95-64) 11l:unused	Section 8.9.1.25
0x74000	pe_rule_chanlut0[512]		PE Channel Rule LUT OBSAI: part0 CPRI: link0	Section 8.9.1.26
0x74800	pe_rule_chanlut1[512]		PE Channel Rule LUT OBSAI: part1 CPRI: link1	Section 8.9.1.27
0x75000	pe_rule_chanlut2[512]		PE Channel Rule LUT OBSAI: part2 CPRI: link2	Section 8.9.1.28
0x75800	pe_rule_chanlut3[512]		PE Channel Rule LUT OBSAI: part3 CPRI: link3	Section 8.9.1.29
0x76000	pe_rule_chanlut4[512]		PE Channel Rule LUT OBSAI: part4 CPRI: link4	Section 8.9.1.30
0x76800	pe_rule_chanlut5[512]		PE Channel Rule LUT OBSAI: part5 CPRI: link5	Section 8.9.1.31
0x77000	pe_rule_chanlut6[512]		PE Channel Rule LUT OBSAI: part6 CPRI: unused	Section 8.9.1.32
0x77800	pe_rule_chanlut7[512]		PE Channel Rule LUT OBSAI: part7 CPRI: unused	Section 8.9.1.33

8.9.1 Link Registers Details

8.9.1.1 pe_link[0] Register (offset = 0x68000)

Table 8-93. pe_link[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD4	NOT_ACCESSIBLE	RESERVED
28-24	delay	READ_WRITE	Delay (in sys_clks) between DB read and PE processing. Gives time for DB RAM fetch prior to message construction. Nominally programmed at 28 for OBSAI and 0 for CPRI
23-21	RSVD3	NOT_ACCESSIBLE	RESERVED
20	obsai_bub_bw	READ_WRITE	OBSAI Only. DBM rule Bubbles are used for traffic. Channel is indicated by 64th location of Channel Rule LUT. Recommended for Pkt traffic only and recommended only for cases where XCNT is less than 64
19-18	RSVD2	NOT_ACCESSIBLE	RESERVED
17-16	lk_rate	READ_WRITE	Link rate (5x na for OBSAI). • 0 = 8X_RATE • 1 = 4x_RATE • 2 = 2X_RATE • 3 = 5X_RATE
15-5	RSVD1	NOT_ACCESSIBLE	RESERVED
4	obsai_cpri	READ_WRITE	OBSAI or CPRI mode of operation. • 0 = CPRI mode • 1 = OBSAI mode
3	gsm_compress	READ_WRITE	(GSM OBSAI only) Supplying DMA data for each GSM Time Slot is not fully matched with the length of Time Slot. AIF2 PE will not consider this an error and will insert OBSAI empty message in the gaps, and will sync up to the beginning of the next GSM Time Slot 0 = GSM Compressed Mode is Off 1 = GSM Compressed Mode is ON; AIF2 inserts OBSAI Empty messages for missing DMA Data
2	tdd_axc	READ_WRITE	AIF2 is tolerant of whole symbols of missing data on AxC-by-AxC basis. In TDD, Software is permitted to have gaps of whole symbols. (Also support for Base Band hopping of GSM antenna carriers. Software manages which antenna each symbol is sent to.)
1	dio_cppi	READ_WRITE	All AxC within the link are treated as DIO traffic. Packets still remain as Multicore Navigator traffic. 0 = Multicore Navigator 1 = DIO
0	lk_en	READ_WRITE	Link Enable Disable

pe_link[1]	**Access = READ_WRITE**	**Address [0x68800]**
pe_link[2]	**Access = READ_WRITE**	**Address [0x69000]**
pe_link[3]	**Access = READ_WRITE**	**Address [0x69800]**
pe_link[4]	**Access = READ_WRITE**	**Address [0x6A000]**
pe_link[5]	**Access = READ_WRITE**	**Address [0x6A800]**

8.9.1.2 pe_crc[0] Register (offset = 0x68004)

Table 8-94. pe_crc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	RSVD	NOT_ACCESSIBLE	RESERVED
15-8	Crc8_seed	READ_WRITE	CRC8: programmable starting seed value (CRC16 and CRC32 seed is fixed)
7-0	Crc8_poly	READ_WRITE	CRC: programmable polynomial for CRC8 (other polynomials are fixed)
pe_crc[1]			**Address [0x68804]**
pe_crc[2]			**Address [0x69004]**
pe_crc[3]			**Address [0x69804]**
pe_crc[4]			**Address [0x6A004]**
pe_crc[5]			**Address [0x6A804]**

8.9.1.3 pe_cpri_dbm[0] Register (offset = 0x68008)

Table 8-95. pe_cpri_dbm[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD1	NOT_ACCESSIBLE	RESERVED
30-24	cpri_dbm_2size	READ_WRITE	DBMF number of bits of bit map2 to use. Set n value
23	RSVD	NOT_ACCESSIBLE	RESERVED
22-16	cpri_dbm_1size	READ_WRITE	DBMF number of bits of bit map1 to use. Set n-1 value
15-11	cpri_dbm_1mult	READ_WRITE	DBMF repetitions of map1. set N-1 value
10-7	cpri_dbm_xbubble	READ_WRITE	DBMF bubble count where 0:indicates 1 bubble of 1 AxC sample. OBSAI DBMF assumes 4 AxC samples as a bubble length. A burst of bubbles is inserted only when dictated to do so by the bit maps. This field gives great flexibility in rate matching
6-0	cpri_dbm_x	READ_WRITE	CPRI Dual Bit Map FSM (DBMF) channel count, sets the max number of AxC supported in a given Link. that is, for CPRI 4x LTE15MHz this must be set to 7b'0000001 indicating 2 AxC

pe_cpri_dbm[1]	**Access = READ_WRITE**	**Address [0x68808]**
pe_cpri_dbm[2]	**Access = READ_WRITE**	**Address [0x69008]**
pe_cpri_dbm[3]	**Access = READ_WRITE**	**Address [0x69808]**
pe_cpri_dbm[4]	**Access = READ_WRITE**	**Address [0x6A008]**
pe_cpri_dbm[5]	**Access = READ_WRITE**	**Address [0x6A808]**

8.9.1.4 pe_cpri_bitswap[0] Register (offset = 0x6800C)

Table 8-96. pe_cpri_bitswap[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	RESERVED
7-4	post_enc_bitswap	READ_WRITE	Swaps bits in bytes after 4B,5B,Null encoders. Bit0 for control channel 0 ~ bit3 for control channel 3. this swaps control, packet data bit order within byte (or 10 bits) AIF2 doesn't support nibble level swapping
3-0	pre_enc_bitswap	READ_WRITE	Swaps bits in bytes before 4B,5B,Null encoders. Bit0 for control channel 0 ~ bit3 for control channel 3. this swaps control, packet data bit order within byte (or 10 bits) AIF2 doesn't support nibble level swapping

pe_cpri_bitswap[1]	**Access = READ_WRITE**	**Address [0x6880C]**
pe_cpri_bitswap[2]	**Access = READ_WRITE**	**Address [0x6900C]**
pe_cpri_bitswap[3]	**Access = READ_WRITE**	**Address [0x6980C]**
pe_cpri_bitswap[4]	**Access = READ_WRITE**	**Address [0x6A00C]**
pe_cpri_bitswap[5]	**Access = READ_WRITE**	**Address [0x6A80C]**

8.9.1.5 pe_cpridbm_1map[0][0-3] Register (offset = 0x68010)

Table 8-97. pe_cpridbm_1map[0][0-3] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	cpri_dbm_1map	READ_WRITE	DBMF bit map1.
pe_cpridbm_1map[1] [0-3]			**Address [0x68810 – 0x6881F]**
pe_cpridbm_1map[2] [0-3]			**Address [0x69010 – 0x6901F]**
pe_cpridbm_1map[3] [0-3]			**Address [0x69810 – 0x6981F]**
pe_cpridbm_1map[4] [0-3]			**Address [0x6A010 - 0x6A01F]**
pe_cpridbm_1map[5] [0-3]			**Address [0x6A810 – 0x6A81F]**

8.9.1.6 pe_cpridbm_2map[0][0-2] Register (offset = 0x68020)

Table 8-98. pe_cpridbm_2map[0][0-2] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	cpri_dbm_2map	READ_WRITE	DBMF bit map2.
pe_cpridbm_2map[1] [0-2]			**Address [0x68820 – 0x6882B]**
pe_cpridbm_2map[2] [0-2]			**Address [0x69020 – 0x6902B]**
pe_cpridbm_2map[3] [0-2]			**Address [0x69820 – 0x6982B]**
pe_cpridbm_2map[4] [0-2]			**Address [0x6A020- 0x6A02B]**
pe_cpridbm_2map[5] [0-2]			**Address [0x6A820 – 0x6A82B]**

8.9.1.7 pe_cpri0[0] Register (offset = 0x6802C)

Table 8-99. pe_cpri0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-30	pkt_delim_ch3	READ_WRITE	CRPI Control Word 4B/5B encoding enable
29-28	pkt_delim_ch2	READ_WRITE	CPRI Control Word & Packet encoding option
27-26	pkt_delim_ch1	READ_WRITE	CPRI Control Word & Packet encoding option
25-24	pkt_delim_ch0	READ_WRITE	CPRI Control Word & Packet encoding option 0 = User defined data is not extracted from CPRI Control Words 1 = CW user data is delimited using 4B/5B encoding where SOP and EOP are defined 2 = CW user data is delimited by a user defined Null Delimiter (defined in other MMR field) 3 = Packet Boundaries are inferred to be on hyper-frame boundaries
23-21	RSVD1	NOT_ACCESSIBLE	RESERVED
20-12	cpri_nulldelm	READ_WRITE	CPRI Control Word Null Delimiter. Used instead of 4B/5B for packet delimitation. Only used when enabled by above fields
11-2	RSVD	NOT_ACCESSIBLE	RESERVED
1-0	cpri_axc_pack	READ_WRITE	CPRI: identifies the bit precision of the IQ data. Used to unpacking packet data passed over CPRI AxC slots 0 = 7 Bit Samples:AxC data is 7bit I and 7bit Q (byte packing needed) 1 = 8 Bit Samples:AxC data is 8bit I and 8bit Q (byte aligned) 2 = 15 Bit Samples:AxC data is 15bit I and 15bit Q (byte packing needed) 3 = 16 Bit Samples:AxC data is 16bit I and 16bit Q (byte aligned)

pe_cpri0[1]	**Access = READ_WRITE**	**Address [0x6882C]**
pe_cpri0[2]	**Access = READ_WRITE**	**Address [0x6902C]**
pe_cpri0[3]	**Access = READ_WRITE**	**Address [0x6982C]**
pe_cpri0[4]	**Access = READ_WRITE**	**Address [0x6A02C]**
pe_cpri0[5]	**Access = READ_WRITE**	**Address [0x6A82C]**

8.9.1.8 pe_cpri1[0] Register (offset = 0x68030)

Table 8-100. pe_cpri1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	cpri_pkt3_ch_en	READ_WRITE	Enables usage of pack3 circuit. When disabled, 0x1, data is dropped on the floor without capture to a DMA channel
30-24	cpri_pkt3_ch	READ_WRITE	When enabled, indicates which DMA channel should be associated with this stream
23	cpri_pkt2_ch_en	READ_WRITE	same... pack2
22-16	cpri_pkt2_ch	READ_WRITE	same... pack2
15	cpri_pkt1_ch_en	READ_WRITE	same... pack1
14-8	cpri_pkt1_ch	READ_WRITE	same... pack1
7	cpri_pkt0_ch_en	READ_WRITE	same... pack0
6-0	cpri_pkt0_ch	READ_WRITE	same... pack0

pe_cpri1[1]	**Access = READ_WRITE**	**Address [0x68830]**
pe_cpri1[2]	**Access = READ_WRITE**	**Address [0x69030]**
pe_cpri1[3]	**Access = READ_WRITE**	**Address [0x69830]**
pe_cpri1[4]	**Access = READ_WRITE**	**Address [0x6A030]**
pe_cpri1[5]	**Access = READ_WRITE**	**Address [0x6A830]**

8.9.1.9 pe_cw_lut[0][0-255] Register (offset = 0x68400)

Table 8-101. pe_cw_lut[0][0-255] Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	RESERVED
2	cw_en	READ_WRITE	All possible CPRI CW per hyperframe. Dictates whether the control word should be captured at all
1-0	cw_chan	READ_WRITE	All possible CPRI CW per hyperframe are mapped to one of four CPRI CW staging areas. This allow CPRI CW to be split into 4 different streams.
pe_cw_lut[1] [0-255]			**Address [0x68c00 – 0x68FFF]**
pe_cw_lut[2] [0-255]			**Address [0x69400 – 0x697FF]**
pe_cw_lut[3] [0-255]			**Address [0x69c00 – 0x69FFF]**
pe_cw_lut[4] [0-255]			**Address [0x6A400 – 0x6A7FF]**
pe_cw_lut[5] [0-255]			**Address [0x6Ac00 – 0x6AFFF]**

8.9.1.10 pe_global Register (offset = 0x6C000)

Table 8-102. pe_global Register Field Descriptions

Bits	Field Name	Type	Description
31-13	RSVD3	NOT_ACCESSIBLE	RESERVED
12	dio_len	READ_WRITE	DirectIO buffer length, set same as value in DB 0 = 128 bytes, normal case used for WCDMA 1 = 256 bytes, future possible use for LTE 20MHz
11-9	RSVD2	NOT_ACCESSIBLE	RESERVED
8	enet_hdr_sel	READ_WRITE	bit order for Ethernet preamble and SOF 0: 0xAAAAAAAB 1: 0x555555D5
7	RSVD1	NOT_ACCESSIBLE	RESERVED
6-4	token_phase	READ_WRITE	Gives Phase alignment relative to Channel Radio Frame Boundary for scheduling DMA. OBSAI: only lsb is used CPRI: all three bits give eighth phase alignment. DIO: has no effect PktDMA: Phase can be used to adjust DMA timing according to the amount of DMA transfer. Normally set to zero.
3-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	shrt_frm_mode	READ_WRITE	Short Frame mode, used for hardware test only. Shortens the OBSAI and CPRI frames for purposes of reducing simulation time. 0 = SHRT_FRM_OFF 1 = SHRT_FRM_ON

8.9.1.11 pe_global_en_set Register (offset = 0x6C004)

Table 8-103. pe_global_en_set Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register will set (enable) global enable

8.9.1.12 pe_global_en_clr (offset = 0x6C008)

Table 8-104. pe_global_en_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register will clear (disable) global enable

8.9.1.13 pe_global_en_sts Register (offset = 0x6C00C)

Table 8-105. pe_global_en_sts Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	enable	READ	0x1: pe_ON 0x0:pe_OFF

8.9.1.14 pe_chan_sts[4] Register (offset = 0x6C010)

Table 8-106. pe_chan_sts[4] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	chan_on	READ	0x1: CHAN_ON 0x0:CHAN_OFF

8.9.1.15 pe_pkt_sts[4] Register (offset = 0x6C020)

Table 8-107. pe_pkt_sts[4] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	pkt_sts	READ	0x1: In Packet 0x0: Out Packet

8.9.1.16 pe_frm_tc[6] Register (offset = 0x6C080)

Table 8-108. pe_frm_tc[6] Register Field Descriptions

Bits	Field Name	Type	Description
31-24	RSVD2	NOT_ACCESSIBLE	RESERVED
23-16	frm_sym_tc	READ_WRITE	Radio framing counter. Symbol terminal count.
15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-8	frm_index_sc	READ_WRITE	Radio framing counter. Index Start count.
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	frm_index_tc	READ_WRITE	Radio framing counter. Index Terminal count.

8.9.1.17 pe_dmachan_en[128] Register (offset = 0x6C200)

Table 8-109. pe_dmachan_en[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ch_en	READ_WRITE	Enable channels one-by-one

8.9.1.18 pe_dma0chan[128] Register (offset = 0x6C400)

Table 8-110. pe_dma0chan[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-17	RSVD3	NOT_ACCESSIBLE	RESERVED
16	crc_hdr	READ_WRITE	CRC calculation for OBSAI header 1 = CRC16 is calculated over 3 byte OBSAI header and 14 bytes of 16 byte payload. Only valid for 19 byte OBSAI messages utilizing CRC16 0 = do not perform CRC over OBSAI Header (program to OFF for CPRI and for most OBSAI types)
15-13	RSVD2	NOT_ACCESSIBLE	RESERVED
12	ethernet	READ_WRITE	Channel is ethernet. This field controls insertion of the Ethernet Premble and SOF 1 = channel is ethernet 0 = channel is not Ethernet
11	RSVD1	NOT_ACCESSIBLE	RESERVED
10-9	crc_type	READ_WRITE	CRC: length of CRC 0 = 32BIT_CRC 1 = 16BIT_CRC 2 = 8BIT_CRC
8-7	rt_ctl	READ_WRITE	Controls RT to perform appropriate insterion/aggregation into PHY 3 = ADD8: WCDMA (Aggregation), OFDM (Insertion) 2 = ADD16: WCDMA (Aggregation), OFDM (Insertion) 1 = INSERTPE: will use PE generated data only 0 = RETRANS: will use Ingress redirected data only
6-4	frm_tc	READ_WRITE	OBSAI framing counter: Which of 6 framing counter terminal counts should be used for each dma channel. (codes 3b110 and 3b111 are reserved)
3-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	crc_en	READ_WRITE	CRC: enable CRC check on AxC by AxC basis

8.9.1.19 pe_in_fifo[128] Register (offset = 0x6C600)

Table 8-111. pe_in_fifo[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-24	RSVD1	NOT_ACCESSIBLE	RESERVED
23-16	sync_sym	READ_WRITE	Normally set to 0. Check first non-zero symbol of frame. Used for startup sync with DMA. If 0th symbol is zero'ed and 1st symbol has data in TDD mode, then this value should be 1
15-8	RSVD	NOT_ACCESSIBLE	RESERVED
7-4	mf_full_lev	READ_WRITE	Message Construction FIFO: Set full level per channel. Full level controls halt of data fetch to prevent overflow.
3-0	mf_wmark	READ_WRITE	Message Construction FIFO: Set water mark per channel. Water mark controls fill level for which lower/higher fetch rate is issued

8.9.1.20 pe_axc_offset[128] Register (offset = 0x6C800)

Table 8-112. pe_axc_offset[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-25	RSVD	NOT_ACCESSIBLE	RESERVED
24-0	axc_offset	READ_WRITE	Antanna carrier offset. Programmed in 307.2 MHz sys clocks from the RadT frame boundary (OBSAI) or 245.76 MHz sys clocks from the phy frame boundary for CPRI.

8.9.1.21 pe_frm_msg_tc[128] Register (offset = 0x6CA00)

Table 8-113. pe_frm_msg_tc[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-16	RSVD	NOT_ACCESSIBLE	RESERVED
15-0	frme_msg_tc	READ_WRITE	Framing message terminal count table OBSAI: number of 16-byte message terminal count. CPRI: number of four byte sample

8.9.1.22 pe_modtxrule[64] Register (offset = 0x6CC00)

Table 8-114. pe_modtxrule[64] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD1	NOT_ACCESSIBLE	RESERVED
30-28	rule_lk	READ_WRITE	Indicates which link the rule is allocated to
27-16	rule_index	READ_WRITE	Index. Transmission Rule. Index indicates at which count the rule will fire. (from 0 to rule_mod)
15-14	RSVD	NOT_ACCESSIBLE	RESERVED
13	rule_ctl_msg	READ_WRITE	Bit 1: Module Rule appellations on OBSAI control messages bit0: AxC messages
12	rule_en	READ_WRITE	Transmission Rule Enable.
11-0	rule_mod	READ_WRITE	Modulo Terminal count of the rule counter. Dictates the period of the rule. Terminal count is the OBSAI Modulo minus 1.

8.9.1.23 pe_obsai_hdr[0-127] Register (offset = 0x6CE00)

Table 8-115. pe_obsai_hdr[0-127] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD	NOT_ACCESSIBLE	RESERVED
30	bb_jop	READ_WRITE	OBSAI Address is taken from Multicore Navigator protocol specific bits (Software value passed in data dma) instead of obsai_adr mmr bits
29	obsai_pkt	READ_WRITE	OBSAI channel (not used for CPRI) is a packet channel not AxC channel. AxC traffic is given higher DMA priority than Pkt traffic
28-26	obsai_ts_frmt	READ_WRITE	TS: OBSAI timestamp generation algorithm 0 = No timestamp check, might be useful for some future radio standard or debug 1 = normal timestamp format 2 = GSM OBSAI UL Timestamp, UL timestamp format msb=1 for first four msg. 3 = Generic Packet (SOP=10, MOP=00, EOP=11) 4 = Ethernet Type where last TS indicates number of valid bytes in last xfer 5 = TS checked by pe_Route, one message packet with inferred SOP and EOP in same OBSAI message 6 = GSM OBSAI DL Timestamp, DL timestamp format msb=1 for first msg
25-24	obsai_ts_mask	READ_WRITE	Controls which parts of the OBSAI TS field are inserted vs. calculated 0 = All TS bits are generated, not inserted 1 = 4 lsb bits of TS are inserted, 2 msb are generated 2 = All TS(5-0) bits are inserted 3 = Reserved
23-11	obsai_adr	READ_WRITE	OBSAI hdr[23-11]ADR: inserted into OBSAI header ADDRESS field. 13 bit unique address to differentiate channels in whole system
10-6	obsai_type	READ_WRITE	OBSAI hdr[10-6] TYPE: inserted int OBSAI header TYPE field
5-0	obsai_ts_adr	READ_WRITE	OBSAI hdr[5-0] TS: only used if TS is inserted instead of generated

8.9.1.24 pe_obsai_dbm[64] Register (offset = 0x70000)

Table 8-116. pe_obsai_dbm[64] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD3	NOT_ACCESSIBLE	RESERVED
30-24	dbm_2size	READ_WRITE	DBMF number of bits of bit map2 to use. Set n value
23	RSVD2	NOT_ACCESSIBLE	RESERVED
22-16	dbm_1size	READ_WRITE	DBMF number of bits of bit map1 to use. Set n-1 value
15-13	RSVD1	NOT_ACCESSIBLE	RESERVED
12-8	dbm_1mult	READ_WRITE	DBMF repetitions of map1 set N-1 value
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	dbm_x	READ_WRITE	DBMF max number of supported channels. set X-1 value

8.9.1.25 pe_dbm_map[512] Register (offset = 0x70800)

Table 8-117. pe_dbm_map[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	dbm_bit_map	READ_WRITE	DBMF bit map 1&2 (0: no bubble 1: with bubble) [4] bitmap 1 plus [4] bitmap 2 (final one array is reserved) and it is repeated for [64] rules

8.9.1.26 pe_rule_chanlut0[512] Register (offset = 0x74000)

Table 8-118. pe_rule_chanlut0[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD	NOT_ACCESSIBLE	RESERVED
8	cpri_pkt_en	READ_WRITE	DBMF CPRI Stream LUT: dictates the cpri payload is to be used as AxC (normal) or Packet traffic. Control channel also need to set this. (note: the concept of packet traffic in CPRI payload is a feature extension of CPRI) 0 = CPRI_AXC 1 = CPRI_PKT
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.9.1.27 pe_rule_chanlut1[512] Register (offset = 0x74800)

Table 8-119. pe_rule_chanlut1[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD	NOT_ACCESSIBLE	RESERVED
8	cpri_pkt_en	READ_WRITE	DBMF CPRI Stream LUT: dictates the cpri payload is to be used as AxC (normal) or Packet traffic. Control channel also need to set this. (note: the concept of packet traffic in CPRI payload is a feature extension of CPRI) 0 = CPRI_AXC 1 = CPRI_PKT
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.9.1.28 pe_rule_chanlut2[512] Register (offset = 0x75000)

Table 8-120. pe_rule_chanlut2[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD	NOT_ACCESSIBLE	RESERVED
8	cpri_pkt_en	READ_WRITE	DBMF CPRI Stream LUT: dictates the cpri payload is to be used as AxC (normal) or Packet traffic. Control channel also need to set this. (note: the concept of packet traffic in CPRI payload is a feature extension of CPRI) 0 = CPRI_AXC 1 = CPRI_PKT
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.9.1.29 pe_rule_chanlut3[512] Register (offset = 0x75800)

Table 8-121. pe_rule_chanlut3[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD	NOT_ACCESSIBLE	RESERVED
8	cpri_pkt_en	READ_WRITE	DBMF CPRI Stream LUT: dictates the cpri payload is to be used as AxC (normal) or Packet traffic. Control channel also need to set this. (note: the concept of packet traffic in CPRI payload is a feature extension of CPRI) 0 = CPRI_AXC 1 = CPRI_PKT
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.9.1.30 pe_rule_chanlut4[512] Register (offset = 0x76000)

Table 8-122. pe_rule_chanlut4[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD	NOT_ACCESSIBLE	RESERVED
8	cpri_pkt_en	READ_WRITE	DBMF CPRI Stream LUT: dictates the cpri payload is to be used as AxC (normal) or Packet traffic. Control channel also need to set this. (note: the concept of packet traffic in CPRI payload is a feature extension of CPRI)) 0 = CPRI_AXC 1 = CPRI_PKT
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.9.1.31 pe_rule_chanlut5[512] Register (offset = 0x76800)

Table 8-123. pe_rule_chanlut5[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-9	RSVD	NOT_ACCESSIBLE	RESERVED
8	cpri_pkt_en	READ_WRITE	DBMF CPRI Stream LUT: dictates the cpri payload is to be used as AxC (normal) or Packet traffic. Control channel also need to set this. (note: the concept of packet traffic in CPRI payload is a feature extension of CPRI) 0 = CPRI_AXC 1 = CPRI_PKT
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.9.1.32 pe_rule_chanlut6[512] Register (offset = 0x77000)

Table 8-124. pe_rule_chanlut6[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	RESERVED
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.9.1.33 pe_rule_chanlut7[512] Register (offset = 0x77800)

Table 8-125. pe_rule_chanlut7[512] Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	RESERVED
7	chanindex_en	READ_WRITE	enable entry
6-0	chanindex	READ_WRITE	DMA channel number. OBSAI: LUT maps rule indexes (modulo and DBMR) to DMA channels. CPRI: LUT maps dbm X index to DMA channels

8.10 DB Registers

Table 8-126. DB Register Memory Map

Offset	Register Name	Description	Section
0x10000	DB_IDB_CFG	Ingress Data Buffer Configuration Register	Section 8.10.1.1
0x10004	DB_IDB_GLOBAL_EN_SET	Set Global Enable for Ingress DB	Section 8.10.1.2
0x10008	DB_IDB_GLOBAL_EN_CLR	Clear Global Enable for Ingress DB	Section 8.10.1.3
0x1000C	DB_IDB_GLOBAL_EN_STS	Read Only status of Ingress DB global enable state.	Section 8.10.1.4
0x10010	DB_IDB_CH_EN[4]	Ingress Data Buffer Channel Enable Register. Enables for 32 contiguous Ingress data buffer channels as follows: DB_IDB_CH_EN[0] -> channels[31-0]; DB_IDB_CH_EN[1] -> channels[63-32]; DB_IDB_CH_EN[2] -> channels[95-64]; DB_IDB_CH_EN[3] -> channels[127-96];	Section 8.10.1.5
0x10100	DB_IDB_DEBUG_D0	Ingress DB Debug Data Register 0	Section 8.10.1.6
0x10104	DB_IDB_DEBUG_D1	Ingress DB Debug Data Register 1	Section 8.10.1.7
0x10108	DB_IDB_DEBUG_D2	Ingress DB Debug Data Register 2	Section 8.10.1.8
0x1010C	DB_IDB_DEBUG_D3	Ingress DB Debug Data Register 3	Section 8.10.1.9
0x10110	DB_IDB_DEBUG_SBND	This register provides sideband data that is written into the Ingress DB RAM during debug.	Section 8.10.1.10
0x10114	DB_IDB_DEBUG_DB_WR	A write of any value to this register writes the data in the following registers into the Ingress DB RAM and sideband RAMS: DB_IDB_DEBUG_D0, DB_IDB_DEBUG_D1, DB_IDB_DEBUG_D2, DB_IDB_DEBUG_D3, DB_IDB_DEBUG_SBDN0, DB_IDB_DEBUG_SBDN1	Section 8.10.1.11
0x10118	DB_IDB_DEBUG_OFS	This register provides the addresses to access the Ingress read and write offset RAMs for debug.	Section 8.10.1.12
0x1011C	DB_IDB_DEBUG_OFS_DAT	This register contains the offset value read from the Ingress read and write offset RAMs for debug.	Section 8.10.1.13
0x10200	DB_IDB_PTR_CH[128]	There is a separate register for each of the 128 Ingress Data Buffer channels. These registers are used to create the read and write addresses to the Ingress DB RAM.	Section 8.10.1.14
0x10400	DB_IDB_CFG_CH[12]	There is a separate register for each of the 128 Ingress Buffer channels. These registers provide configuration data for each channel when it is read.	Section 8.10.1.15
0x10600	DB_IDB_CH_EMPTY[4]	Ingress Data Buffer Channel Empty Register. Empty bits for 32 contiguous Ingress data buffer channels as follows: DB_IDB_CH_EMPTY[0] -> channels[31:0]; DB_IDB_CH_EMPTY[1] -> channels[63:32]; DB_IDB_CH_EMPTY[2] -> channels[95:64]; DB_IDB_CH_EMPTY[3] -> channels[127:96];	Section 8.10.1.16
0x11000	DB_EDB_CFG	Egress Data Buffer Configuration Register	Section 8.10.1.17
0x11004	DB_EDB_GLOBAL_EN_SET	Set Global Enable for Egress DB	Section 8.10.1.18
0x11008	DB_EDB_GLOBAL_EN_CLR	Clear Global Enable for Egress DB	Section 8.10.1.19
0x1100C	DB_EDB_GLOBAL_EN_STS	Read Only status of Egress DB global enable state.	Section 8.10.1.20
0x11010	DB_EDB_CH_EN[4]	Egress Data Buffer Channel Enable Register. Enables for 32 contiguous Egress data buffer channels as follows: DB_EDB_CH_EN[0] -> channels[31-0]; DB_EDB_CH_EN[1] -> channels[63-32]; DB_EDB_CH_EN[2] -> channels[95-64]; DB_EDB_CH_EN[3] -> channels[127-96];	Section 8.10.1.21
0x11100	DB_EDB_DEBUG_D0	Egress DB Debug Data Register 0.	Section 8.10.1.22
0x11104	DB_EDB_DEBUG_D1	Egress DB Debug Data Register 1.	Section 8.10.1.23
0x11108	DB_EDB_DEBUG_D2	Egress DB Debug Data Register 2.	Section 8.10.1.24
0x1110C	DB_EDB_DEBUG_D3	Egress DB Debug Data Register 3.	Section 8.10.1.25
0x11110	DB_EDB_DEBUG_SBND	This register provides part of the sideband data that is written into the Egress DB RAM during debug.	Section 8.10.1.26
0x11114	DB_EDB_DEBUG_RD_CNTL	This register provides the controls necessary to read the Egress DB RAM during debug.	Section 8.10.1.27

Table 8-126. DB Register Memory Map (continued)

Offset	Register Name	Description	Section
0x1111C	DB_EDB_DEBUG_DB_RD	A write of any value to this register initiates a read to the Egress DB data and sideband RAMs and writes the contents into the following registers: DB_EDB_DEBUG_D0, DB_EDB_DEBUG_D1, DB_EDB_DEBUG_D2, DB_EDB_DEBUG_D3, DB_EDB_DEBUG_SBND	Section 8.10.1.28
0x11120	DB_EDB_DEBUG_OFS	This register provides the addresses to access the Egress read and write offset RAMs for debug.	Section 8.10.1.29
0x11124	DB_EDB_DEBUG_OFS_DAT	This register contains the offset value read from the Egress read and write offset RAMs for debug.	Section 8.10.1.30
0x11128	DB_EDB_DEBUG_WR_TOK	A write of any value to this register results in a token with the value loaded into DB_EDB_DEBUG_RD_CNTL.CH_ID being issued to the AxC Token FIFO	Section 8.10.1.31
0x1112C	DB_EDB_EOP_CNT	This register provides a count of the Egress EOPs received from the PKTDMA Controller for activity monitoring.	Section 8.10.1.32
0x11200	DB_EDB_PTR_CH[128]	There is a separate register for each of the 128 Egress Data Buffer channels. These registers are used to create the read and write addresses to the Egress DB RAM.	Section 8.10.1.33
0x11400	DB_EDB_CFG_CH[128]	There is a separate register for each of the 128 Egress Buffer channels. These registers provide configuration data for each channel when it is read.	Section 8.10.1.34

8.10.1 Grouped Common Registers Details

8.10.1.1 DB_IDB_CFG Register (offset = 0x10000)

Table 8-127. DB_IDB_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD2	NOT_ACCESSIBLE	RESERVED
7	DT_EN	READ_WRITE	Data Trace Enable 0 = Clear data trace control logic in IDB and AD. This has the effect of flushing the contents of the Data Trace RAM and the Data Trace Token FIFO 1 = Enable capture of trace data and/or framing data per settings of DTB_EN and DTF_EN
6	DT_SYNC	READ_WRITE	Data Trace Sync Enable. When data trace capture is synchronized to the frame boundary, the least significant byte of the first quad-word is always the first byte of the frame regardless of whether the first byte was the even or odd byte 0 = capture data trace data without any synchronization 1 = synchronize start of data capture to frame boundary signal from RM
5	DTF_EN	READ_WRITE	Trace Framing Data Capture Enable 0 = do not capture trace framing data 1 = capture trace framing data
4	DTB_EN	READ_WRITE	Trace Data Capture Enable 0 = do not capture trace data 1 = capture trace data
3-2	RSVD	NOT_ACCESSIBLE	RESERVED
1	DIO_LEN	READ_WRITE	Direct IO buffer length 0 = 128 bytes 1 = 256 bytes
0	IDB_DEBUG_EN	READ_WRITE	Enable Ingress DB debug mode 0 = Ingress DB not in debug mode 1 = Ingress DB in debug mode

8.10.1.2 DB_IDB_GLOBAL_EN_SET Register (offset = 0x10004)

Table 8-128. DB_IDB_GLOBAL_EN_SET Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register sets the global enable for the Ingress DB

8.10.1.3 DB_IDB_GLOBAL_EN_CLR Register (offset = 0x10008)

Table 8-129. DB_IDB_GLOBAL_EN_CLR Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register clears the global enable for the Ingress DB

8.10.1.4 DB_IDB_GLOBAL_EN_STS Register (offset = 0x1000C)

Table 8-130. DB_IDB_GLOBAL_EN_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ENABLE	READ	0x1: Ingress DB_ON 0x0:Ingress DB OFF

8.10.1.5 DB_IDB_CH_EN[4] Register (offset = 0x10010)

Table 8-131. DB_IDB_CH_EN[4] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	EN	READ_WRITE	Each bit is an enable for 1 of 32 contiguous Ingress data buffer channels

8.10.1.6 DB_IDB_DEBUG_D0 Register (offset = 0x10100)

Table 8-132. DB_IDB_DEBUG_D0 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ_WRITE	Debug data written to bits 31-0 of Ingress DB RAM

8.10.1.7 DB_IDB_DEBUG_D1 Register (offset = 0x10104)

Table 8-133. DB_IDB_DEBUG_D1 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ_WRITE	Debug data written to bits 63-32 of Ingress DB RAM

8.10.1.8 DB_IDB_DEBUG_D2 Register (offset = 0x10108)

Table 8-134. DB_IDB_DEBUG_D2 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ_WRITE	Debug data written to bits 95-64 of Ingress DB RAM

8.10.1.9 DB_IDB_DEBUG_D3 Register (offset = 0x1010C)

Table 8-135. DB_IDB_DEBUG_D3 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ_WRITE	Debug data written to bits 127-96 of Ingress DB RAM

8.10.1.10 DB_IDB_DEBUG_SBND Register (offset = 0x10110)

Table 8-136. DB_IDB_DEBUG_SBND Register Field Descriptions

Bits	Field Name	Type	Description
31-24	symbol	READ_WRITE	AxC symbol number inserted as part of protocol specific data (0x00 - 0xFF)
23-21	RSVD1	NOT_ACCESSIBLE	RESERVED
20-16	xcnt	READ_WRITE	Transfer Byte Count: Indicates how many valid bytes are transferred in the current data phase. 0 = no valid bytes 1 = 1 valid byte 2 = 2 valid bytes 3 = 3 valid bytes 4 = 4 valid bytes 5 = 5 valid bytes 6 = 6 valid bytes 7 = 7 valid bytes 8 = 8 valid bytes 9 = 9 valid bytes 10 = 10 valid bytes 11 = 11 valid bytes 12 = 12 valid bytes 13 = 13 valid bytes 14 = 14 valid bytes 15 = 15 valid bytes 16 = 16 valid bytes
15-12	dio_addr	READ_WRITE	Address within a Direct IO Circular Buffer. Each address contains a quad-word and 16 locations support up to 256 byte circular buffers.
11	RSVD	NOT_ACCESSIBLE	RESERVED
10-4	ch_id	READ_WRITE	DMA channel number 127-0
3	eop	READ_WRITE	End of packet 0 = not end of packet 1 = end of packet
2	sop	READ_WRITE	Start of packet 0 = not start of packet 1 = start of packet
1	fifo_wr_en	READ_WRITE	FIFO Buffer Write Enable 0 = FIFO buffer write disabled 1 = FIFO buffer write enabled
0	dio_wr_en	READ_WRITE	Direct IO Buffer Write Enable 0 = Direct IO buffer write disabled 1 = Direct IO buffer write enabled

8.10.1.11 DB_IDB_DEBUG_DB_WR Register (offset = 0x10114)

Table 8-137. DB_IDB_DEBUG_DB_WR Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register writes the data in the following registers into the Ingress DB and sideband RAMS: DB_IDB_DEBUG_D0, DB_IDB_DEBUG_D1, DB_IDB_DEBUG_D2, DB_IDB_DEBUG_D3, DB_IDB_DEBUG_SBDN0, DB_IDB_DEBUG_SBDN1

8.10.1.12 DB_IDB_DEBUG_OFS Register (offset = 0x10118)

Table 8-138. DB_IDB_DEBUG_OFS Register Field Descriptions

Bits	Field Name	Type	Description
31-15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-8	raddr	READ_WRITE	Address used to access read offset RAM
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	waddr	READ_WRITE	Address used to access write offset RAM

8.10.1.13 DB_IDB_DEBUG_OFS_DAT Register (offset = 0x1011C)

Table 8-139. DB_IDB_DEBUG_OFS_DAT Register Field Descriptions

Bits	Field Name	Type	Description
31-16	RSVD1	NOT_ACCESSIBLE	RESERVED
15-8	roff	READ	Read offset value at address in DB_IDB_DEBUG_OFS.RADDR
7-0	woff	READ	Write Offset Value at address in DB_IDB_DEBUG_OFS.WADDR

8.10.1.14 DB_IDB_PTR_CH[128] Register (offset = 0x10200)

Table 8-140. DB_IDB_PTR_CH[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-11	RSVD1	NOT_ACCESSIBLE	RESERVED
10-8	BUF_DEPTH	READ_WRITE	Depth of FIFO Buffer in number of quad-words (ignored for Direct IO circular buffers) 0 = 8 quad-words 1 = 16 quad-words 2 = 32 quad-words 3 = 64 quad-words 4 = 128 quad-words 5 = 256 quad-words 6 = 256 quad-words 7 = 256 quad-words
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	BASE_ADDR	READ_WRITE	Starting address of per-channel FIFO or Direct IO buffer.

8.10.1.15 DB_IDB_CFG_CH[128] Register (offset = 0x10400)

Table 8-141. DB_IDB_CFG_CH[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-21	RSVD3	NOT_ACCESSIBLE	RESERVED
20-16	PKT_TYPE	READ_WRITE	Programmable packet type that is inserted into pkt_type field in Multicore Navigator Info Word 0.
15-9	RSVD2	NOT_ACCESSIBLE	RESERVED
8	PS_EN	READ_WRITE	Protocol Specific Data Enable. 0 = Do not prepend protocol specific data onto packet 1 = Prepend packet protocol specific data onto packet
7-6	RSVD1	NOT_ACCESSIBLE	RESERVED
5-4	IQ_ORDER	READ_WRITE	IQ swapping scheme. 0 = no swap 1 = no swap 2 = byte swap 3 = 16-bit swap
3-2	RSVD	NOT_ACCESSIBLE	RESERVED
1-0	DAT_SWAP	READ_WRITE	Big endian swapping control. 0 = no swap 1 = byte swap 2 = half word swap. 16-bit swap 3 = word swap. 32-bits

8.10.1.16 DB_IDB_CH_EMPTY[4] Register (offset = 0x10600)

Table 8-142. DB_IDB_CH_EMPTY[4] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	EMPTY	READ	Each bit is a buffer empty bit for 1 of 32 contiguous Ingress data buffer channels. Rx Channel tearing down requires this register to check FIFO status for each channel. reset value is 0xFFFFFFFF

8.10.1.17 DB_EDB_CFG Register (offset = 0x11000)

Table 8-143. DB_EDB_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	RESERVED
2	PM_CTL	READ_WRITE	Packet mode control 0 = Put PM tokens from PE in separate PM Token FIFO 1 = Put PM tokens from PE in Axc Token FIFO to improve CPRI packet performance. This mode could be used for non-AxC packet transfer for CPRI.
1	DIO_LEN	READ_WRITE	Direct IO buffer length 0 = 128 bytes 1 = 256 bytes
0	EDB_DEBUG_EN	READ_WRITE	Enable Egress DB debug mode 0 = Egress DB not in debug mode 1 = Engress DB in debug mode

8.10.1.18 DB_EDB_GLOBAL_EN_SET Register (offset = 0x11004)

Table 8-144. DB_EDB_GLOBAL_EN_SET Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register sets the global enable for the Egress DB

8.10.1.19 DB_EDB_GLOBAL_EN_CLR Register (offset = 0x11008)

Table 8-145. DB_EDB_GLOBAL_EN_CLR Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register clears the global enable for the Egress DB

8.10.1.20 DB_EDB_GLOBAL_EN_STS Register (offset = 0x1100C)

Table 8-146. DB_EDB_GLOBAL_EN_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ENABLE	READ	0x1: Egress DB_ON 0x0:Egress DB OFF

8.10.1.21 DB_EDB_CH_EN[4] Register (offset = 0x11010)

Table 8-147. DB_EDB_CH_EN[4] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	EN	READ_WRITE	Each bit is an enable for 1 of 32 contiguous Egress data buffer channels

8.10.1.22 DB_EDB_DEBUG_D0 Register (offset = 0x11100)

Table 8-148. DB_EDB_DEBUG_D0 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ	Debug data read from bits 31-0 of Egress DB RAM

8.10.1.23 DB_EDB_DEBUG_D1 Register (offset = 0x11104)

Table 8-149. DB_EDB_DEBUG_D1 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ	Debug data read from bits 63-32 of Egress DB RAM

8.10.1.24 DB_EDB_DEBUG_D2 Register (offset = 0x11108)

Table 8-150. DB_EDB_DEBUG_D2 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ	Debug data read from bits 95-64 of Egress DB RAM

8.10.1.25 DB_EDB_DEBUG_D3 Register (offset = 0x1110C)

Table 8-151. DB_EDB_DEBUG_D3 Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DATA	READ	Debug data read from bits 127-96 of Egress DB RAM

8.10.1.26 DB_EDB_DEBUG_SBND Register (offset = 0x11110)

Table 8-152. DB_EDB_DEBUG_SBND Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD3	NOT_ACCESSIBLE	RESERVED
30-24	axc	READ	AxC number from protocol specific data. Only valid when SOP asserted. Not applicable for Direct IO.
23-16	symbol	READ	SYMBOL NUMBER from protocol specific data. Only valid when SOP asserted. Not applicable for Direct IO.
15-9	RSVD2	NOT_ACCESSIBLE	RESERVED
8-4	xcnt	READ	Transfer Byte Count: Indicates how many bytes in DB_EDB_DEBUG_D0, DB_EDB_DEBUG_D1, DB_EDB_DEBUG_D2, DB_EDB_DEBUG_D3 and DB_EDB_DEBUG_SBND are valid. 0 = no valid bytes 1 = 1 valid byte 2 = 2 valid bytes 3 = 3 valid bytes 4 = 4 valid bytes 5 = 5 valid bytes 6 = 6 valid bytes 7 = 7 valid bytes 8 = 8 valid bytes 9 = 9 valid bytes 10 = 10 valid bytes 11 = 11 valid bytes 12 = 12 valid bytes 13 = 13 valid bytes 14 = 14 valid bytes 15 = 15 valid bytes 16 = 16 valid bytes
3-2	RSVD1	NOT_ACCESSIBLE	RESERVED
1	eop	READ	End of packet 0 = not end of packet 1 = end of packet
0	sop	READ	Start of packet 0 = not start of packet 1 = start of packet

8.10.1.27 DB_EDB_DEBUG_RD_CNTL Register (offset = 0x11114)

Table 8-153. DB_EDB_DEBUG_RD_CNTL Register Field Descriptions

Bits	Field Name	Type	Description
31-11	RSVD1	NOT_ACCESSIBLE	RESERVED
10-4	ch_id	READ_WRITE	Egress Data Buffer channel number 127-0; Also used as token channel number when DB_EDB_DEBUG_WR_TOK written
3-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	dio_rd_en	READ_WRITE	Direct IO Buffer Read Enable 0 = Direct IO buffer read disabled (Debug reads are for FIFO buffers) 1 = Direct IO buffer read enabled

8.10.1.28 DB_EDB_DEBUG_DB_RD Register (offset = 0x1111C)

Table 8-154. DB_EDB_DEBUG_DB_RD Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register initiates a read to the Egress DB data and sideband RAMs and writes the contents into the following registers: DB_EDB_DEBUG_D0, DB_EDB_DEBUG_D1, DB_EDB_DEBUG_D2, DB_EDB_DEBUG_D3, DB_EDB_DEBUG_SBND

8.10.1.29 DB_EDB_DEBUG_OFS Register (offset = 0x11120)

Table 8-155. DB_EDB_DEBUG_OFS Register Field Descriptions

Bits	Field Name	Type	Description
31-15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-8	raddr	READ_WRITE	Address used to access read offset RAM. The address value is matched with channel number
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	waddr	READ_WRITE	Address used to access write offset RAM. The address value is matched with channel number

8.10.1.30 DB_EDB_DEBUG_OFS_DAT Register (offset = 0x11124)

Table 8-156. DB_EDB_DEBUG_OFS_DAT Register Field Descriptions

Bits	Field Name	Type	Description
31-16	RSVD	NOT_ACCESSIBLE	RESERVED
15-8	roff	READ	Read offset value at address in DB_EDB_DEBUG_OFS.RADDR
7-0	woff	READ	Write Offset Value at address in DB_EDB_DEBUG_OFS.WADDR

8.10.1.31 DB_EDB_DEBUG_WR_TOK Register (offset = 0x11128)

Table 8-157. DB_EDB_DEBUG_WR_TOK Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register results in a token with the value loaded into DB_EDB_DEBUG_RD_CNTL.CH_ID being issued to the AxC Token FIFO

8.10.1.32 DB_EDB_EOP_CNT Register (offset = 0x1112C)

Table 8-158. DB_EDB_EOP_CNT Register Field Descriptions

Bits	Field Name	Type	Description
31-24	RSVD	NOT_ACCESSIBLE	RESERVED
23-0	eop_cnt	READ	Wrapping count of EOPs received from the PKTDMA Controller.

8.10.1.33 DB_EDB_PTR_CH[128] Register (offset = 0x11200)

Table 8-159. DB_EDB_PTR_CH[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-11	RSVD1	NOT_ACCESSIBLE	RESERVED
10-8	BUF_DEPTH	READ_WRITE	Depth of FIFO Buffer in number of quad-words (ignored for Direct IO circular buffers) 0 = 8 quad-words 1 = 16 quad-words 2 = 32 quad-words 3 = 64 quad-words 4 = 128 quad-words 5 = 256 quad-words 6 = 256 quad-words 7 = 256 quad-words
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	BASE_ADDR	READ_WRITE	Starting address of per-channel FIFO or Direct IO buffer.

8.10.1.34 DB_EDB_CFG_CH[128] Register (offset = 0x11400)

Table 8-160. DB_EDB_CFG_CH[128] Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD2	NOT_ACCESSIBLE	RESERVED
11-8	dio_offset	READ_WRITE	Offset for DIO DMA mode. Used to align DMA for AxC with different AxC offsets. Zero means zero offset and when it is increased one, the actual offset will be increased by 4 or 8 WCDMA chips (DL, UL)
7-6	RSVD1	NOT_ACCESSIBLE	RESERVED
5-4	IQ_ORDER	READ_WRITE	IQ swapping scheme. 0 = no swap 1 = no swap 2 = byte swap 3 = 16-bit swap
3-2	RSVD	NOT_ACCESSIBLE	RESERVED
1-0	DAT_SWAP	READ_WRITE	Big endian swapping control. 0 = no swap 1 = byte swap 2 = half word swap. 16-bit swap 3 = word swap. 32-bits

8.11 AD Registers

Table 8-161. AD Register Memory Map

Offset	Register Name	Access	Description	Section
0xC000	AD_DIO_I_TABLE_SEL[0]	R/W	AD DIO Ingress Table Select Register (I DIO Engine 0)	Section 8.11.1.1
0xC004	AD_DIO_I_TABLE_LOOP_CFG[0]	R/W	AD DIO Ingress Table Loop Configuration Register	Section 8.11.1.2
0xC008	AD_DIO_I_DMA_CFG0[0]	R/W	AD DIO Ingress DMA Configuration Register 0	Section 8.11.1.3
0xC00C	AD_DIO_I_DMA_CFG1[0]	R/W	AD DIO Ingress DMA Configuration Register 1	Section 8.11.1.4
0xC010	AD_DIO_I_DMA_CFG2[0]	R/W	AD DIO Ingress DMA Configuration Register 2	Section 8.11.1.5
0xC014 to 0xC050	AD_DIO_I_BCN_TABLE0_ROW0 to AD_DIO_I_BCN_TABLE0_ROW15[0]	R/W	AD DIO Ingress BCN Table 0 Row 0 ~ Row15 Register	Section 8.11.1.6
0xC054 to 0xC090	AD_DIO_I_BCN_TABLE1_ROW0 to AD_DIO_I_BCN_TABLE1_ROW15[0]	R/W	AD DIO Ingress BCN Table 1 Row 0 ~ Row 15 Register	Section 8.11.1.7
0xC100	AD_DIO_I_TABLE_SEL[1]	R/W	AD DIO Ingress Table Select Register (I DIO Engine 1)	Section 8.11.1.1
0xC104	AD_DIO_I_TABLE_LOOP_CFG[1]	R/W	AD DIO Ingress Table Loop Configuration Register	Section 8.11.1.2
0xC108	AD_DIO_I_DMA_CFG0[1]	R/W	AD DIO Ingress DMA Configuration Register 0	Section 8.11.1.3
0xC10C	AD_DIO_I_DMA_CFG1[1]	R/W	AD DIO Ingress DMA Configuration Register 1	Section 8.11.1.4
0xC110	AD_DIO_I_DMA_CFG2[1]	R/W	AD DIO Ingress DMA Configuration Register 2	Section 8.11.1.5
0xC114 to 0xC150	AD_DIO_I_BCN_TABLE0_ROW0 to AD_DIO_I_BCN_TABLE0_ROW15[1]	R/W	AD DIO Ingress BCN Table 0 Row 0 ~ Row15 Register	Section 8.11.1.6
0xC154 to 0xC190	AD_DIO_I_BCN_TABLE1_ROW0 to AD_DIO_I_BCN_TABLE1_ROW15[1]	R/W	AD DIO Ingress BCN Table 1 Row 0 ~ Row 15 Register	Section 8.11.1.7
0xC200	AD_DIO_I_TABLE_SEL[2]	R/W	AD DIO Ingress Table Select Register (I DIO Engine 2)	Section 8.11.1.1
0xC204	AD_DIO_I_TABLE_LOOP_CFG[2]	R/W	AD DIO Ingress Table Loop Configuration Register	Section 8.11.1.2
0xC208	AD_DIO_I_DMA_CFG0[2]	R/W	AD DIO Ingress DMA Configuration Register 0	Section 8.11.1.3
0xC20C	AD_DIO_I_DMA_CFG1[2]	R/W	AD DIO Ingress DMA Configuration Register 1	Section 8.11.1.4
0xC210	AD_DIO_I_DMA_CFG2[2]	R/W	AD DIO Ingress DMA Configuration Register 2	Section 8.11.1.5
0xC214 to 0xC250	AD_DIO_I_BCN_TABLE0_ROW0 to AD_DIO_I_BCN_TABLE0_ROW15[2]	R/W	AD DIO Ingress BCN Table 0 Row 0 ~ Row15 Register	Section 8.11.1.6
0xC254 to 0xC290	AD_DIO_I_BCN_TABLE1_ROW0 to AD_DIO_I_BCN_TABLE1_ROW15[2]	R/W	AD DIO Ingress BCN Table 1 Row 0 ~ Row 15 Register	Section 8.11.1.7
0xC300	AD_DIO_E_TABLE_SEL[0]	R/W	AD DIO Egress Table Select Register (E DIO Engine 0)	Section 8.11.1.8
0xC304	AD_DIO_E_TABLE_LOOP_CFG[0]	R/W	AD DIO Egress Table Loop Configuration Register	Section 8.11.1.9
0xC308	AD_DIO_E_DMA_CFG0[0]	R/W	AD DIO Egress DMA Configuration Register 0	Section 8.11.1.10
0xC30C	AD_DIO_E_DMA_CFG1[0]	R/W	AD DIO Egress DMA Configuration Register 1	Section 8.11.1.11
0xC310	AD_DIO_E_DMA_CFG2[0]	R/W	AD DIO Egress DMA Configuration Register 2	Section 8.11.1.12
0xC314 to 0xC350	AD_DIO_E_BCN_TABLE0_ROW0 to AD_DIO_E_BCN_TABLE0_ROW15[0]	R/W	AD DIO Egress BCN Table 0 Row 0 ~ Row15 Register	Section 8.11.1.13

Table 8-161. AD Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0xC354 to 0xC390	AD_DIO_E_BCN_TABLE1_ROW0 to AD_DIO_E_BCN_TABLE1_ROW15[0]	R/W	AD DIO Egress BCN Table 1 Row 0 ~ Row 15 Register	Section 8.11.1.14
0xC400	AD_DIO_E_TABLE_SEL[1]	R/W	AD DIO Egress Table Select Register (E DIO Engine 1)	Section 8.11.1.8
0xC404	AD_DIO_E_TABLE_LOOP_CFG[1]	R/W	AD DIO Egress Table Loop Configuration Register	Section 8.11.1.9
0xC408	AD_DIO_E_DMA_CFG0[1]	R/W	AD DIO Egress DMA Configuration Register 0	Section 8.11.1.10
0xC40C	AD_DIO_E_DMA_CFG1[1]	R/W	AD DIO Egress DMA Configuration Register 1	Section 8.11.1.11
0xC410	AD_DIO_E_DMA_CFG2[1]	R/W	AD DIO Egress DMA Configuration Register 2	Section 8.11.1.12
0xC414 to 0xC454	AD_DIO_E_BCN_TABLE0_ROW0 to AD_DIO_E_BCN_TABLE0_ROW15[1]	R/W	AD DIO Egress BCN Table 0 Row 0 ~ Row15 Register	Section 8.11.1.13
0xC418 to 0xC490	AD_DIO_E_BCN_TABLE1_ROW0 to AD_DIO_E_BCN_TABLE1_ROW15[1]	R/W	AD DIO Egress BCN Table 1 Row 0 ~ Row 15 Register	Section 8.11.1.14
0xC500	AD_DIO_E_TABLE_SEL[2]	R/W	AD DIO Egress Table Select Register (E DIO Engine 2)	Section 8.11.1.8
0xC504	AD_DIO_E_TABLE_LOOP_CFG[2]	R/W	AD DIO Egress Table Loop Configuration Register	Section 8.11.1.9
0xC508	AD_DIO_E_DMA_CFG0[2]	R/W	AD DIO Egress DMA Configuration Register 0	Section 8.11.1.10
0xC50C	AD_DIO_E_DMA_CFG1[2]	R/W	AD DIO Egress DMA Configuration Register 1	Section 8.11.1.11
0xC510	AD_DIO_E_DMA_CFG2[2]	R/W	AD DIO Egress DMA Configuration Register 2	Section 8.11.1.12
0xC514 to 0xC550	AD_DIO_E_BCN_TABLE0_ROW0 to AD_DIO_E_BCN_TABLE0_ROW15[2]	R/W	AD DIO Egress BCN Table 0 Row 0 ~ Row15 Register	Section 8.11.1.13
0xC518 to 0xC590	AD_DIO_E_BCN_TABLE1_ROW0 to AD_DIO_E_BCN_TABLE1_ROW15[2]	R/W	AD DIO Egress BCN Table 1 Row 0 ~ Row 15 Register	Section 8.11.1.14
0xC600	AD_DIO_DT_DMA_CFG0		AD DIO Data Trace DMA Configuration Register 0	Section 8.11.1.15
0xC604	AD_DIO_DT_DMA_CFG1		AD DIO Data Trace DMA Configuration Register 1	Section 8.11.1.16
0xC608	AD_DIO_DT_DMA_CFG2		AD DIO Data Trace DMA Configuration Register 2	Section 8.11.1.17
0xC60C	AD_DIO_DT_DMA_CFG3		AD DIO Data Trace DMA Configuration Register 3	Section 8.11.1.18
0xC610	AD_DIO_I_GLOBAL_EN_SET		Set Global Enable for AD Ingress DIO	Section 8.11.1.19
0xC614	AD_DIO_I_GLOBAL_EN_CLR		Clear Global Enable for AD Ingress DIO	Section 8.11.1.20
0xC618	AD_DIO_I_GLOBAL_EN_STS		Read Only status of AD Ingress DIO global enable state	Section 8.11.1.21
0xC61C	AD_DIO_E_GLOBAL_EN_SET		Set Global Enable for AD Egress DIO	Section 8.11.1.22
0xC620	AD_DIO_E_GLOBAL_EN_CLR		Clear Global Enable for AD Egress DIO	Section 8.11.1.23
0xC624	AD_DIO_E_GLOBAL_EN_STS		Read Only status of AD Egress DIO global enable state	Section 8.11.1.24
0xE000	AD_ISCH_CFG		AD Scheduler Ingress Configuration Register	Section 8.11.1.25
0xE004	AD_ISCH_GLOBAL_EN_SET		Set Global Enable for AD Ingress Scheduler	Section 8.11.1.26
0xE008	AD_ISCH_GLOBAL_EN_CLR		Clear Global Enable for AD Ingress Scheduler	Section 8.11.1.27
0xE00C	AD_ISCH_GLOBAL_EN_STS		Read Only status of AD Ingress Scheduler global enable state	Section 8.11.1.28

Table 8-161. AD Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0xE010	AD_ISCH_EOP_CNT		This register provides a count of the EOPs sent to the PKTDMA Controller for activity monitoring	Section 8.11.1.29
0xE100	AD_ESCH_CFG		AD Scheduler Egress Configuration Register	Section 8.11.1.30
0xE104	AD_ESCH_GLOBAL_EN_SET		Set Global Enable for AD Egress Scheduler	Section 8.11.1.31
0xE108	AD_ESCH_GLOBAL_EN_CLR		Clear Global Enable for AD Egress Scheduler	Section 8.11.1.32
0xE10C	AD_ESCH_GLOBAL_EN_STS		Read Only status of AD Egress Scheduler global enable state	Section 8.11.1.33

8.11.1 Grouped Common Registers Details

8.11.1.1 AD_DIO_I_TABLE_SEL[0] Register (offset = 0xC000)

Table 8-162. AD_DIO_I_TABLE_SEL[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	bcn_table_sel	READ_WRITE	Selects which buffer channel number table for the DMA engine to use 0 = Selects buffer channel number table 0 1 = Selects buffer channel number table 1
AD_DIO_I_TABLE_SEL[1]		**Access = READ_WRITE**	**Address [0xC100]**
AD_DIO_I_TABLE_SEL[2]		**Access = READ_WRITE**	**Address [0xC200]**

8.11.1.2 AD_DIO_I_TABLE_LOOP_CFG[0] Register (offset = 0xC004)

Table 8-163. AD_DIO_I_TABLE_LOOP_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-14	RSVD1	NOT_ACCESSIBLE	RESERVED
13-8	num_axc	READ_WRITE	Sets the number of AxCs (that is, set 0 to 63 for 1 to 64 AxCs).
7-2	RSVD	NOT_ACCESSIBLE	RESERVED
1-0	num_qw	READ_WRITE	Sets the number of quad words per AxC. 0 = 1 quad word per AxC 1 = 2 quad word per AxC 2 = 4 quad word per AxC
AD_DIO_I_TABLE_LOOP_CFG[1]		**Access = READ_WRITE**	**Address [0xC104]**
AD_DIO_I_TABLE_LOOP_CFG[2]		**Access = READ_WRITE**	**Address [0xC204]**

8.11.1.3 AD_DIO_I_DMA_CFG0[0] Register (offset = 0xC008)

Table 8-164. AD_DIO_I_DMA_CFG0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD2	NOT_ACCESSIBLE	RESERVED
28-16	dma_num_blks	READ_WRITE	Set the number of data blocks to transfer before wrapping back to dma_base_addr.(Set N-1 value)
15-9	RSVD1	NOT_ACCESSIBLE	RESERVED
8	dma_ch_en	READ_WRITE	Enable channel DMA. 0 = DMA channel disabled that is, cleared state 1 = DMA Channel enabled
7-2	RSVD	NOT_ACCESSIBLE	RESERVED
1-0	dma_brst_ln	READ_WRITE	Sets the maximum DMA burst length. 0 = 1 quad word transferred per burst 1 = 2 quad words transferred per burst 2 = 4 quad words transferred per burst
AD_DIO_I_DMA_CFG0[1]		**Access = READ_WRITE**	**Address [0xC108]**
AD_DIO_I_DMA_CFG0[2]		**Access = READ_WRITE**	**Address [0xC208]**

8.11.1.4 AD_DIO_I_DMA_CFG1[0] Register (offset = 0xC00C)

Table 8-165. AD_DIO_I_DMA_CFG1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	NOT_ACCESSIBLE	RESERVED
27-0	dma_base_addr	READ_WRITE	Sets the VBUS destination base address (upper 28 bits of 32-bit data bus).
AD_DIO_I_DMA_CFG1[1]		**Access = READ_WRITE**	**Address [0xC10C]**
AD_DIO_I_DMA_CFG1[2]		**Access = READ_WRITE**	**Address [0xC20C]**

8.11.1.5 AD_DIO_I_DMA_CFG2[0] Register (offset = 0xC010)

Table 8-166. AD_DIO_I_DMA_CFG2[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD1	NOT_ACCESSIBLE	RESERVED
27-16	dma_blk_addr_stride	READ_WRITE	Sets the DMA block address stride (in multiples of 0x10).
15-12	RSVD	NOT_ACCESSIBLE	RESERVED
11-0	dma_brst_addr_stride	READ_WRITE	Sets the DMA burst address stride (in multiples of 0x10). After each DMA burst, the DMA address will increment by this amount.
AD_DIO_I_DMA_CFG2[1]		**Access = READ_WRITE**	**Address [0xC110]**
AD_DIO_I_DMA_CFG2[2]		**Access = READ_WRITE**	**Address [0xC210]**

8.11.1.6 AD_DIO_I_BCN_TABLE0_ROW0[0] to AD_DIO_I_BCN_TABLE0_ROW15[0] Register (offset = 0xC014 to 0xC050)

Table 8-167. AD_DIO_I_BCN_TABLE0_ROW0[0] to AD_DIO_I_BCN_TABLE0_ROW15[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD3	NOT_ACCESSIBLE	RESERVED
30-24	dbcn3 ~ dbcn63	READ_WRITE	Data Buffer Channel Number 3 (~ 63)
23	RSVD2	NOT_ACCESSIBLE	RESERVED
22-16	dbcn2 ~ dbcn62	READ_WRITE	Data Buffer Channel Number 2 (~ 62)
15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-8	dbcn1 ~ dbcn61	READ_WRITE	Data Buffer Channel Number 1 (~ 61)
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	dbcn0 ~ dbcn60	READ_WRITE	Data Buffer Channel Number 0 (~ 60)
AD_DIO_I_BCN_TABLE0_ROW0[1] ~ AD_DIO_I_BCN_TABLE0_ROW15[1]		**Access = READ_WRITE**	**Address [0xC114] ~ [0xC150]**
AD_DIO_I_BCN_TABLE0_ROW0[2] ~ AD_DIO_I_BCN_TABLE0_ROW15[2]		**Access = READ_WRITE**	**Address [0xC214] ~ [0xC250]**

8.11.1.7 AD_DIO_I_BCN_TABLE1_ROW0[0] to AD_DIO_I_BCN_TABLE1_ROW15[0] Register (offset = 0xC054 to 0xC090)

Table 8-168. AD_DIO_I_BCN_TABLE1_ROW0[0] to AD_DIO_I_BCN_TABLE1_ROW15[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD3	NOT_ACCESSIBLE	RESERVED
30-24	dbcn3 ~ dbcn63	READ_WRITE	Data Buffer Channel Number 3 (~ 63)
23	RSVD2	NOT_ACCESSIBLE	RESERVED
22-16	dbcn2 ~ dbcn62	READ_WRITE	Data Buffer Channel Number 2 (~ 62)
15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-8	dbcn1 ~ dbcn61	READ_WRITE	Data Buffer Channel Number 1 (~ 61)
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	dbcn0 ~ dbcn60	READ_WRITE	Data Buffer Channel Number 0 (~ 60)
AD_DIO_I_BCN_TABLE1_ROW0[1] AD_DIO_I_BCN_TABLE1_ROW15[1]		**Access = READ_WRITE**	**Address [0xC154] ~ [0xC190]**
AD_DIO_I_BCN_TABLE1_ROW0[2] AD_DIO_I_BCN_TABLE1_ROW15[2]		**Access = READ_WRITE**	**Address [0xC254] ~ [0xC290]**

8.11.1.8 AD_DIO_E_TABLE_SEL[0] Register (offset = 0xC300)

Table 8-169. AD_DIO_E_TABLE_SEL[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	bcn_table_sel	READ_WRITE	Selects which buffer channel number table for the DMA engine to use 0 = Selects buffer channel number table 0 1 = Selects buffer channel number table 1
AD_DIO_E_TABLE_SEL[1]		**Access = READ_WRITE**	**Address [0xC400]**
AD_DIO_E_TABLE_SEL[2]		**Access = READ_WRITE**	**Address [0xC500]**

8.11.1.9 AD_DIO_E_TABLE_LOOP_CFG[0] Register (offset = 0xC304)

Table 8-170. AD_DIO_E_TABLE_LOOP_CFG[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-14	RSVD1	NOT_ACCESSIBLE	RESERVED
13-8	num_axc	READ_WRITE	Sets the number of AxCs (that is, set 0 to 63 for 1 to 64 AxCs). 0 = 1 quad word per AxC 1 = 2 quad word per AxC 2 = 4 quad word per AxC
7-2	RSVD	NOT_ACCESSIBLE	RESERVED
1-0	num_qw	READ_WRITE	Sets the number of quad words per AxC.
AD_DIO_E_TABLE_LOOP_CFG[1]		**Access = READ_WRITE**	**Address [0xC404]**
AD_DIO_E_TABLE_LOOP_CFG[2]		**Access = READ_WRITE**	**Address [0xC504]**

8.11.1.10 AD_DIO_E_DMA_CFG0[0] Register (offset = 0xC308)

Table 8-171. AD_DIO_E_DMA_CFG0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	RESERVED 0 = 1 quad word transferred per burst 1 = 2 quad words transferred per burst 2 = 4 quad words transferred per burst
28-16	dma_num_blks	READ_WRITE	Set the number of data blocks to transfer before wrapping back to dma_base_addr. (set N-1)
15-9	RSVD2	NOT_ACCESSIBLE	RESERVED
8	dma_ch_en	READ_WRITE	Enable channel DMA. 0 = DMA channel disabled that is, cleared state 1 = DMA Channel enabled
7-5	RSVD1	NOT_ACCESSIBLE	RESERVED
4	rsa_en	READ_WRITE	Configure channel DMA for RSA data format 0 = Channel DMA not configured for RSA data format 1 = Channel DMA configured for RSA data format
3-2	RSVD	NOT_ACCESSIBLE	RESERVED
1-0	dma_brst_ln	READ_WRITE	Sets the maximum DMA burst length.
AD_DIO_E_DMA_CFG0[1]		**Access = READ_WRITE**	**Address [0xC408]**
AD_DIO_E_DMA_CFG0[2]		**Access = READ_WRITE**	**Address [0xC508]**

8.11.1.11 AD_DIO_E_DMA_CFG1[0] Register (offset = 0xC30C)

Table 8-172. AD_DIO_E_DMA_CFG1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	NOT_ACCESSIBLE	RESERVED
27-0	dma_base_addr	READ_WRITE	Sets the VBUS destination base address (upper 28 bits of 32-bit data bus).
AD_DIO_E_DMA_CFG1[1]		**Access = READ_WRITE**	**Address [0xC40C]**
AD_DIO_E_DMA_CFG1[2]		**Access = READ_WRITE**	**Address [0xC50C]**

8.11.1.12 AD_DIO_E_DMA_CFG2[0] Register (offset = 0xC310)

Table 8-173. AD_DIO_E_DMA_CFG2[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD1	NOT_ACCESSIBLE	RESERVED
27-16	dma_blk_addr_stride	READ_WRITE	Sets the DMA block address stride (in multiples of 0x10).
15-12	RSVD	NOT_ACCESSIBLE	RESERVED
11-0	dma_brst_addr_stride	READ_WRITE	Sets the DMA burst address stride (in multiples of 0x10). After each DMA burst, the DMA address will increment by this amount.
AD_DIO_E_DMA_CFG2[1]		**Access = READ_WRITE**	**Address [0xC410]**
AD_DIO_E_DMA_CFG2[2]		**Access = READ_WRITE**	**Address [0xC510]**

8.11.1.13 AD_DIO_E_BCN_TABLE0_ROW0[0] to AD_DIO_E_BCN_TABLE0_ROW15[0] Register (offset = 0xC314 to 0xC350)

Table 8-174. AD_DIO_E_BCN_TABLE0_ROW0[0] to AD_DIO_E_BCN_TABLE0_ROW15[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD3	NOT_ACCESSIBLE	RESERVED
30-24	dbcn3 ~ dbcn63	READ_WRITE	Data Buffer Channel Number 3 (~ 63)
23	RSVD2	NOT_ACCESSIBLE	RESERVED
22-16	dbcn2 ~ dbcn62	READ_WRITE	Data Buffer Channel Number 2 (~ 62)
15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-8	dbcn1 ~ dbcn61	READ_WRITE	Data Buffer Channel Number 1 (~ 61)
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	dbcn0 ~ dbcn60	READ_WRITE	Data Buffer Channel Number 0 (~ 60)
AD_DIO_E_BCN_TABLE0_ROW0[1] ~ AD_DIO_E_BCN_TABLE0_ROW15[1]		**Access = READ_WRITE**	**Address [0xC414] ~ [0xC450]**
AD_DIO_E_BCN_TABLE0_ROW0[2] ~ AD_DIO_E_BCN_TABLE0_ROW15[2]		**Access = READ_WRITE**	**Address [0xC514] ~ [0xC550]**

8.11.1.14 AD_DIO_E_BCN_TABLE1_ROW0[0] to AD_DIO_E_BCN_TABLE1_ROW15[0] Register (offset = 0xC354 to 0xC390)

Table 8-175. AD_DIO_E_BCN_TABLE1_ROW0[0] to AD_DIO_E_BCN_TABLE1_ROW15[0] Register Field Descriptions

Bits	Field Name	Type	Description
31	RSVD3	NOT_ACCESSIBLE	RESERVED
30-24	dbcn3 ~ dbcn63	READ_WRITE	Data Buffer Channel Number 3 (~ 63)
23	RSVD2	NOT_ACCESSIBLE	RESERVED
22-16	dbcn2 ~ dbcn62	READ_WRITE	Data Buffer Channel Number 2 (~ 62)
15	RSVD1	NOT_ACCESSIBLE	RESERVED
14-8	dbcn1 ~ dbcn61	READ_WRITE	Data Buffer Channel Number 1 (~ 61)
7	RSVD	NOT_ACCESSIBLE	RESERVED
6-0	dbcn0 ~ dbcn60	READ_WRITE	Data Buffer Channel Number 0 (~ 60)
AD_DIO_E_BCN_TABLE1_ROW0[1] ~ AD_DIO_E_BCN_TABLE1_ROW15[1]		**Access = READ_WRITE**	**Address [0xC454] ~ [0xC490]**
AD_DIO_E_BCN_TABLE1_ROW0[2] ~ AD_DIO_E_BCN_TABLE1_ROW15[2]		**Access = READ_WRITE**	**Address [0xC554] ~ [0xC590]**

8.11.1.15 AD_DIO_DT_DMA_CFG0 Register (offset = 0xC600)

Table 8-176. AD_DIO_DT_DMA_CFG0 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	RESERVED
1	dt_dma_fm_ch_en	READ_WRITE	Enable data trace framing dtaa channel DMA. 0 = DMA channel disabled that is, cleared state 1 = DMA Channel enabled
0	dt_dma_rd_ch_en	READ_WRITE	Enable data trace receive data channel DMA. 0 = DMA channel disabled that is, cleared state 1 = DMA Channel enabled

8.11.1.16 AD_DIO_DT_DMA_CFG1 Register (offset = 0xC604)

Table 8-177. AD_DIO_DT_DMA_CFG1 Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	NOT_ACCESSIBLE	RESERVED
27-0	dt_dma_rd_base_addr	READ_WRITE	Sets the destination VBUS base address (upper 28 bits of 32-bit data bus) for data trace receive data.

8.11.1.17 AD_DIO_DT_DMA_CFG2 Register (offset = 0xC608)

Table 8-178. AD_DIO_DT_DMA_CFG2 Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	NOT_ACCESSIBLE	RESERVED
27-0	dt_dma_fm_base_addr	READ_WRITE	Sets the destination VBUS base address (upper 28 bits of 32-bit data bus) for data trace framing data.

8.11.1.18 AD_DIO_DT_DMA_CFG3n Register (offset = 0xC60C)

Table 8-179. AD_DIO_DT_DMA_CFG3n Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	RESERVED
17-0	dt_dma_wrap	READ_WRITE	Sets the number of burst transfers before the destination address wraps back to the base address. Burst size is 64 bytes for data and 16 bytes for framing data

8.11.1.19 AD_DIO_I_GLOBAL_EN_SET Register (offset = 0xC610)

Table 8-180. AD_DIO_I_GLOBAL_EN_SET Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register sets the global enable for the AD Ingress DIO

8.11.1.20 AD_ DIO_I _GLOBAL_EN_CLR Register (offset = 0xC614)

Table 8-181. AD_ DIO_I _GLOBAL_EN_CLR Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register clears the global enable for the AD Ingress DIO

8.11.1.21 AD_ DIO_I _GLOBAL_EN_STS Register (offset = 0xC618)

Table 8-182. AD_ DIO_I _GLOBAL_EN_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ENABLE	READ	0x1: AD Ingress Scheduler_ON 0x0:AD Ingress DIO OFF

8.11.1.22 AD_DIO_E_GLOBAL_EN_SET Register (offset = 0xC61C)

Table 8-183. AD_DIO_E_GLOBAL_EN_SET Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register sets the global enable for the AD Egress DIO

8.11.1.23 AD_ DIO_E _GLOBAL_EN_CLR Register (offset = 0xC620)

Table 8-184. AD_ DIO_E _GLOBAL_EN_CLR Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register clears the global enable for the AD Egress DIO

8.11.1.24 AD_ DIO_E_GLOBAL_EN_STS Register (offset = 0xC624)

Table 8-185. AD_ DIO_E_GLOBAL_EN_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ENABLE	READ	0x1: AD Ingress Scheduler_ON 0x0:AD Egress DIO OFF

8.11.1.25 AD_ISCH_CFG Register (offset = 0xE000)

Table 8-186. AD_ISCH_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-17	RSVD1	NOT_ACCESSIBLE	RESERVED
16	pri	READ_WRITE	This field controls the Ingress Scheduler arbitration priority 0 = Direct IO requests have highest priority 1 = Packet requests have highest priority
15-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	fail_mode	READ_WRITE	This field controls how the Ingress Scheduler handles packets marked as failed by the PD. 0 = PD drops failed packets (not transferred to PKTDMA) 1 = PD mark failed packets. Transferred to PKTDMA with error mark in the descriptor

8.11.1.26 AD_ISCH_GLOBAL_EN_SET Register (offset = 0xE004)

Table 8-187. AD_ISCH_GLOBAL_EN_SET Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register sets the global enable for the AD Ingress Scheduler

8.11.1.27 AD_ISCH_GLOBAL_EN_CLR Register (offset = 0xE008)

Table 8-188. AD_ISCH_GLOBAL_EN_CLR Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register clears the global enable for the AD Ingress Scheduler

8.11.1.28 AD_ISCH_GLOBAL_EN_STS Register (offset = 0xE00C)

Table 8-189. AD_ISCH_GLOBAL_EN_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ENABLE	READ	0x1: AD Ingress Scheduler_ON 0x0:AD Ingress Scheduler OFF

8.11.1.29 AD_ISCH_EOP_CNT Register (offset = 0xE010)

Table 8-190. AD_ISCH_EOP_CNT Register Field Descriptions

Bits	Field Name	Type	Description
31-24	RSVD	NOT_ACCESSIBLE	RESERVED
23-0	Eop_cnt	READ	Wrapping count of EOPs sent to the PKTDMA COntroller

8.11.1.30 AD_ESCH_CFG Register (offset = 0xE100)

Table 8-191. AD_ESCH_CFG Register Field Descriptions

Bits	Field Name	Type	Description
31-17	RSVD1	NOT_ACCESSIBLE	RESERVED
16	pri	READ_WRITE	This field controls the Egress Scheduler arbitration priority 0 = AxC requests (DIO and Multicore Navigator packet based) have higher priority than non-AxC packet requests. 1 = Should be Used for OBSAI and could be used for CPRI if there's no non-AxC packet
15-4	txq_qnum	READ_WRITE	Base egress queue number for AIF2. Channel id of each packet 127-0 added to this base value to generate egress queue number sent to PKTDMA Scheduling Interface. Normally it is set to 512
3-2	RSVD	NOT_ACCESSIBLE	RESERVED Non-AxC packet requests have higher priority than AxC requests (DIO and Multicore Navigator packet based) requests 1 = Should be used for CPRI if there's non-AxC packet
1-0	txq_qmgr	READ_WRITE	Queue manager number sent to PKTDMA Scheduling Interface for all egress transactions.

8.11.1.31 AD_ESCH_GLOBAL_EN_SET Register (offset = 0xE104)

Table 8-192. AD_ESCH_GLOBAL_EN_SET Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register sets the global enable for the AD Egress Scheduler

8.11.1.32 AD_ESCH_GLOBAL_EN_CLR Register (offset = 0xE108)

Table 8-193. AD_ESCH_GLOBAL_EN_CLR Register Field Descriptions

Bits	Field Name	Type	Description
31-0	DONT_CARE	WRITE	A write of any value to this register clears the global enable for the AD Egress Scheduler

8.11.1.33 AD_ESCH_GLOBAL_EN_STS Register (offset = 0xE10C)

Table 8-194. AD_ESCH_GLOBAL_EN_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	RESERVED
0	ENABLE	READ	0x1: AD Egress Scheduler_ON 0x0:AD Ingress Scheduler OFF

8.12 AT Registers

Table 8-195. AT Register Memory Map

Offset	Register Name	Access	Description	Section
0x48100	at_pimax_lk[0]	R/W	at_pimax_lk for link0	Section 8.12.1.1
0x48104	at_pimin_lk[0]	R/W	at_pimin_lk for link0	Section 8.12.1.2
0x48108	at_pivalue_lk[0]	R	at_pivalue_lk for link0	Section 8.12.1.3
0x4810C	at_pimax_lk[1]	R/W	at_pimax_lk for link1	Section 8.12.1.1
0x48110	at_pimin_lk[1]	R/W	at_pimin_lk for link1	Section 8.12.1.2
0x48114	at_pivalue_lk[1]	R	at_pivalue_lk for link1	Section 8.12.1.3
0x48118	at_pimax_lk[2]	R/W	at_pimax_lk for link2	Section 8.12.1.1
0x4811C	at_pimin_lk[2]	R/W	at_pimin_lk for link2	Section 8.12.1.2
0x48120	at_pivalue_lk[2]	R	at_pivalue_lk for link2	Section 8.12.1.3
0x48124	at_pimax_lk[3]	R/W	at_pimax_lk for link3	Section 8.12.1.1
0x48128	at_pimin_lk[3]	R/W	at_pimin_lk for link3	Section 8.12.1.2
0x4812C	at_pivalue_lk[3]	R	at_pivalue_lk for link3	Section 8.12.1.3
0x48130	at_pimax_lk[4]	R/W	at_pimax_lk for link4	Section 8.12.1.1
0x48134	at_pimin_lk[4]	R/W	at_pimin_lk for link4	Section 8.12.1.2
0x48138	at_pivalue_lk[4]	R	at_pivalue_lk for link4	Section 8.12.1.3
0x4813C	at_pimax_lk[5]	R/W	at_pimax_lk for link5	Section 8.12.1.1
0x48140	at_pimin_lk[5]	R/W	at_pimin_lk for link5	Section 8.12.1.2
0x48144	at_pivalue_lk[5]	R	at_pivalue_lk for link5	Section 8.12.1.3
0x48200	at_event_offset[0]	R/W	at_event0_offset	Section 8.12.1.4
0x48204	at_event_mod_tc[0]	R/W	at_event0_mod_tc	Section 8.12.1.5
0x48208	at_event_mask_lsbs[0]	R/W	at_event0_mask_lsbs	Section 8.12.1.6
0x4820C	at_event_mask_msbs[0]	R/W	at_event0_mask_msbs	Section 8.12.1.7
0x48210	at_event_offset[1]	R/W	at_event1_offset	Section 8.12.1.4
0x48214	at_event_mod_tc[1]	R/W	at_event1_mod_tc	Section 8.12.1.5
0x48218	at_event_mask_lsbs[1]	R/W	at_event1_mask_lsbs	Section 8.12.1.6
0x4821C	at_event_mask_msbs[1]	R/W	at_event1_mask_msbs	Section 8.12.1.7
0x48220	at_event_offset[2]	R/W	at_event2_offset	Section 8.12.1.4
0x48224	at_event_mod_tc[2]	R/W	at_event2_mod_tc	Section 8.12.1.5
0x48228	at_event_mask_lsbs[2]	R/W	at_event2_mask_lsbs	Section 8.12.1.6
0x4822C	at_event_mask_msbs[2]	R/W	at_event2_mask_msbs	Section 8.12.1.7
0x48230	at_event_offset[3]	R/W	at_event3_offset	Section 8.12.1.4
0x48234	at_event_mod_tc[3]	R/W	at_event3_mod_tc	Section 8.12.1.5
0x48238	at_event_mask_lsbs[3]	R/W	at_event3_mask_lsbs	Section 8.12.1.6
0x4823C	at_event_mask_msbs[3]	R/W	at_event3_mask_msbs	Section 8.12.1.7
0x48240	at_event_offset[4]	R/W	at_event4_offset	Section 8.12.1.4
0x48244	at_event_mod_tc[4]	R/W	at_event4_mod_tc	Section 8.12.1.5
0x48248	at_event_mask_lsbs[4]	R/W	at_event4_mask_lsbs	Section 8.12.1.6
0x4824C	at_event_mask_msbs[4]	R/W	at_event4_mask_msbs	Section 8.12.1.7
0x48250	at_event_offset[5]	R/W	at_event5_offset	Section 8.12.1.4
0x48254	at_event_mod_tc[5]	R/W	at_event5_mod_tc	Section 8.12.1.5
0x48258	at_event_mask_lsbs[5]	R/W	at_event5_mask_lsbs	Section 8.12.1.6
0x4825C	at_event_mask_msbs[5]	R/W	at_event5_mask_msbs	Section 8.12.1.7
0x48260	at_event_offset[6]	R/W	at_event6_offset	Section 8.12.1.4
0x48264	at_event_mod_tc[6]	R/W	at_event6_mod_tc	Section 8.12.1.5
0x48268	at_event_mask_lsbs[6]	R/W	at_event6_mask_lsbs	Section 8.12.1.6

Table 8-195. AT Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x4826C	at_event_mask_msbs[6]	R/W	at_event6_mask_msbs	Section 8.12.1.7
0x48270	at_event_offset[7]	R/W	at_event7_offset	Section 8.12.1.4
0x48274	at_event_mod_tc[7]	R/W	at_event7_mod_tc	Section 8.12.1.5
0x48278	at_event_mask_lsbs[7]	R/W	at_event7_mask_lsbs	Section 8.12.1.6
0x4827C	at_event_mask_msbs[7]	R/W	at_event7_mask_msbs	Section 8.12.1.7
0x48280	at_event_offset[8]	R/W	at_event8_offset	Section 8.12.1.4
0x48284	at_event_mod_tc[8]	R/W	at_event8_mod_tc	Section 8.12.1.5
0x48288	at_event_mask_lsbs[8]	R/W	at_event8_mask_lsbs	Section 8.12.1.6
0x4828C	at_event_mask_msbs[8]	R/W	at_event8_mask_msbs	Section 8.12.1.7
0x48290	at_event_offset[9]	R/W	at_event9_offset	Section 8.12.1.4
0x48294	at_event_mod_tc[9]	R/W	at_event9_mod_tc	Section 8.12.1.5
0x48298	at_event_mask_lsbs[9]	R/W	at_event9_mask_lsbs	Section 8.12.1.6
0x4829C	at_event_mask_msbs[9]	R/W	at_event9_mask_msbs	Section 8.12.1.7
0x482A0	at_event_offset[10]	R/W	at_event10_offset	Section 8.12.1.4
0x482A4	at_event_mod_tc[10]	R/W	at_event10_mod_tc	Section 8.12.1.5
0x482A8	at_event_mask_lsbs[10]	R/W	at_event10_mask_lsbs	Section 8.12.1.6
0x482AC	at_event_mask_msbs[10]	R/W	at_event10_mask_msbs	Section 8.12.1.7
0x48400	at_ad_ingr_event_offset[0]	R/W	at_ad_ingr_event0_offset for DIO Engine0	Section 8.12.1.8
0x48404	at_ad_ingr_event_mod_tc[0]	R/W	at_ad_ingr_event0_mod_tc for DIO Engine0	Section 8.12.1.9
0x48408	at_ad_ingr_event_offset[1]	R/W	at_ad_ingr_event1_offset for DIO Engine1	Section 8.12.1.8
0x4840C	at_ad_ingr_event_mod_tc[1]	R/W	at_ad_ingr_event1_mod_tc for DIO Engine1	Section 8.12.1.9
0x48410	at_ad_ingr_event_offset[2]	R/W	at_ad_ingr_event2_offset for DIO Engine2	Section 8.12.1.8
0x48414	at_ad_ingr_event_mod_tc[2]	R/W	at_ad_ingr_event2_mod_tc for DIO Engine2	Section 8.12.1.9
0x48418	at_ad_ingr_event_offset[3]	R/W	at_ad_ingr_event3_offset for DIO Engine 0 frame event	Section 8.12.1.8
0x4841C	at_ad_ingr_event_mod_tc[3]	R/W	at_ad_ingr_event3_mod_tc for DIO Engine 0 frame event	Section 8.12.1.9
0x48420	at_ad_ingr_event_offset[4]	R/W	at_ad_ingr_event4_offset for DIO Engine 1 frame event	Section 8.12.1.8
0x48424	at_ad_ingr_event_mod_tc[4]	R/W	at_ad_ingr_event4_mod_tc for DIO Engine 1 frame event	Section 8.12.1.9
0x48428	at_ad_ingr_event_offset[5]	R/W	at_ad_ingr_event5_offset for DIO Engine 2 frame event	Section 8.12.1.8
0x4842C	at_ad_ingr_event_mod_tc[5]	R/W	at_ad_ingr_event5_mod_tc for DIO Engine 2 frame event	Section 8.12.1.9
0x48500	at_ad_egr_event_offset[0]	R/W	at_ad_egr_event0_offset for DIO Engine0	Section 8.12.1.10
0x48504	at_ad_egr_event_mod_tc[0]	R/W	at_ad_egr_event0_mod_tc for DIO Engine0	Section 8.12.1.11
0x48508	at_ad_egr_event_offset[1]	R/W	at_ad_egr_event1_offset for DIO Engine1	Section 8.12.1.10
0x4850C	at_ad_egr_event_mod_tc[1]	R/W	at_ad_egr_event1_mod_tc for DIO Engine1	Section 8.12.1.11
0x48510	at_ad_egr_event_offset[2]	R/W	at_ad_egr_event2_offset for DIO Engine2	Section 8.12.1.10
0x48514	at_ad_egr_event_mod_tc[2]	R/W	at_ad_egr_event2_mod_tc for DIO Engine2	Section 8.12.1.11

Table 8-195. AT Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x48518	at_ad_egr_event_offset[3]	R/W	at_ad_egr_event3_offset for DIO Engine 0 frame event	Section 8.12.1.10
0x4851C	at_ad_egr_event_mod_tc[3]	R/W	at_ad_egr_event3_mod_tc for DIO Engine 0 frame event	Section 8.12.1.11
0x48520	at_ad_egr_event_offset[4]	R/W	at_ad_egr_event4_offset for DIO Engine 1frame event	Section 8.12.1.10
0x48524	at_ad_egr_event_mod_tc[4]	R/W	at_ad_egr_event4_mod_tc for DIO Engine 1frame event	Section 8.12.1.11
0x48528	at_ad_egr_event_offset[5]	R/W	at_ad_egr_event5_offset for DIO Engine 2frame event	Section 8.12.1.10
0x4852C	at_ad_egr_event_mod_tc[5]	R/W	at_ad_egr_event5_mod_tc for DIO Engine 2frame event	Section 8.12.1.11
0x48540	at_tm_delta_event_offset[0]	R/W	at_tm_delta_event0_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.12
0x48544	at_tm_delta_event_mod_tc[0]	R/W	at_tm_delta_event0_mod_tc configures the modulus terminal count	Section 8.12.1.13
0x48548	at_tm_delta_event_offset[1]	R/W	at_tm_delta_event1_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.12
0x4854C	at_tm_delta_event_mod_tc[1]	R/W	at_tm_delta_event1_mod_tc configures the modulus terminal count	Section 8.12.1.13
0x48550	at_tm_delta_event_offset[2]	R/W	at_tm_delta_event2_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.12
0x48554	at_tm_delta_event_mod_tc[2]	R/W	at_tm_delta_event2_mod_tc configures the modulus terminal count	Section 8.12.1.13
0x48558	at_tm_delta_event_offset[3]	R/W	at_tm_delta_event3_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.12
0x4855C	at_tm_delta_event_mod_tc[3]	R/W	at_tm_delta_event3_mod_tc configures the modulus terminal count	Section 8.12.1.13
0x48560	at_tm_delta_event_offset[4]	R/W	at_tm_delta_event4_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.12
0x48564	at_tm_delta_event_mod_tc[4]	R/W	at_tm_delta_event4_mod_tc configures the modulus terminal count	Section 8.12.1.13
0x48568	at_tm_delta_event_offset[5]	R/W	at_tm_delta_event5_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.12
0x4856C	at_tm_delta_event_mod_tc[5]	R/W	at_tm_delta_event5_mod_tc configures the modulus terminal count	Section 8.12.1.13
0x48580	at_pe_event_offset[0]	R/W	at_pe1_event0_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.14
0x48584	at_pe_event_mod_tc[0]	R/W	at_pe1_event0_mod_tc configures the modulus terminal count	Section 8.12.1.15
0x48588	at_pe_event_offset[1]	R/W	at_pe1_event1_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.14
0x4858C	at_pe_event_mod_tc[1]	R/W	at_pe1_event1_mod_tc configures the modulus terminal count	Section 8.12.1.15
0x48590	at_pe_event_offset[2]	R/W	at_pe1_event2_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.14
0x48594	at_pe_event_mod_tc[2]	R/W	at_pe1_event2_mod_tc configures the modulus terminal count	Section 8.12.1.15

Table 8-195. AT Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x48598	at_pe_event_offset[3]	R/W	at_pe1_event3_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.14
0x4859C	at_pe_event_mod_tc[3]	R/W	at_pe1_event3_mod_tc configures the modulus terminal count	Section 8.12.1.15
0x485A0	at_pe_event_offset[4]	R/W	at_pe1_event4_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.14
0x485A4	at_pe_event_mod_tc[4]	R/W	at_pe1_event4_mod_tc configures the modulus terminal count	Section 8.12.1.15
0x485A8	at_pe_event_offset[5]	R/W	at_pe1_event5_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.14
0x485AC	at_pe_event_mod_tc[5]	R/W	at_pe1_event5_mod_tc configures the modulus terminal count	Section 8.12.1.15
0x485B0	at_pe_event2_offset[0]	R/W	at_pe2_event0_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.16
0x485B4	at_pe_event2_mod_tc[0]	R/W	at_pe2_event0_mod_tc configures the modulus terminal count	Section 8.12.1.17
0x485B8	at_pe_event2_offset[1]	R/W	at_pe2_event1_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.16
0x485BC	at_pe_event2_mod_tc[1]	R/W	at_pe2_event1_mod_tc configures the modulus terminal count	Section 8.12.1.17
0x485C0	at_pe_event2_offset[2]	R/W	at_pe2_event2_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.16
0x485C4	at_pe_event2_mod_tc[2]	R/W	at_pe2_event2_mod_tc configures the modulus terminal count	Section 8.12.1.17
0x485C8	at_pe_event2_offset[3]	R/W	at_pe2_event3_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.16
0x485CC	at_pe_event2_mod_tc[3]	R/W	at_pe2_event3_mod_tc configures the modulus terminal count	Section 8.12.1.17
0x485D0	at_pe_event2_offset[4]	R/W	at_pe2_event4_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.16
0x485D4	at_pe_event2_mod_tc[4]	R/W	at_pe2_event4_mod_tc configures the modulus terminal count	Section 8.12.1.17
0x485D8	at_pe_event2_offset[5]	R/W	at_pe2_event5_offset sets the event offset. Strobe is always PHYT Frame boundary	Section 8.12.1.16
0x485DC	at_pe_event2_mod_tc[5]	R/W	at_pe2_event5_mod_tc configures the modulus terminal count	Section 8.12.1.17
0x48000	at_control1		at_control1	Section 8.12.1.18
0x48004	at_control2		at_control2	Section 8.12.1.19
0x48008	at_sw_sync		at sw sync	Section 8.12.1.20
0x4800C	at_phyt_cmp_radsync		at phyt radsync compare	Section 8.12.1.21
0x48010	at_rp1_type		at rp1 type	Section 8.12.1.22
0x48014	at_captradt		Capture RADT clock and symbol and 5bit frame lsb counts on PHYT frame boundary	Section 8.12.1.23
0x48020	at_rp1_type_capture		at rp1 type capture	Section 8.12.1.24
0x48024	at_rp1_tod_capture_l		at_rp1_tod_capture_l	Section 8.12.1.25
0x48028	at_rp1_tod_capture_h		at_rp1_tod_capture_h	Section 8.12.1.26

Table 8-195. AT Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x4802C	at_rp1_rp3_capture_l		at_rp1_rp3_capture_l	Section 8.12.1.27
0x48030	at_rp1_rp3_capture_h		at_rp1_rp3_capture_h	Section 8.12.1.28
0x48034	at_rp1_rad_capture_l		at_rp1_rad_capture_l	Section 8.12.1.29
0x48038	at_rp1_rad_capture_h		at_rp1_rad_capture_h	Section 8.12.1.30
0x48040	at_phyt_clkcnt_value		at_phyt_clkcnt_value	Section 8.12.1.31
0x48044	at_phyt_frm_value_lsbs		at_phyt_frm_value_lsbs	Section 8.12.1.32
0x48048	at_phyt_frm_value_msbs		at_phyt_frm_value_msbs	Section 8.12.1.33
0x4804C	at_radt_value_lsbs		at_radt_value_lsbs	Section 8.12.1.34
0x48050	at_radt_value_mid		at_radt_value_mid	Section 8.12.1.35
0x48054	at_radt_value_msbs		at_radt_value_msbs	Section 8.12.1.36
0x48058	at_ulradt_value_lsbs		at_ulradt_value_lsbs	Section 8.12.1.37
0x4805C	at_ulradt_value_mid		at_ulradt_value_mid	Section 8.12.1.38
0x48060	at_ulradt_value_msbs		at_ulradt_value_msbs	Section 8.12.1.39
0x48064	at_dlradt_value_lsbs		at_dlradt_value_lsbs	Section 8.12.1.40
0x48068	at_dlradt_value_mid		at_dlradt_value_mid	Section 8.12.1.41
0x4806C	at_dlradt_value_msbs		at_dlradt_value_msbs	Section 8.12.1.42
0x48070	at_radt_wcdma_value		at_radt_wcdma_value	Section 8.12.1.43
0x48074	at_ulradt_wcdma_value		at_ulradt_wcdma_value	Section 8.12.1.44
0x48078	at_dlradt_wcdma_value		at_dlradt_wcdma_value	Section 8.12.1.45
0x4807C	at_radt_wcdma_div		at radt wcdma clock divider terminal count.	Section 8.12.1.46
0x48080	at_phyt_init_lsbs		at_phyt_init_lsbs	Section 8.12.1.47
0x48084	at_phyt_init_mid		at_phyt_init_mid	Section 8.12.1.48
0x48088	at_phyt_init_msbs		at_phyt_init_msbs	Section 8.12.1.49
0x4808C	at_phyt_tc_lsbs		at_phyt_tc_lsbs	Section 8.12.1.50
0x48090	at_phyt_frame_tc_lsbs		at_phyt_frame_tc_lsbs	Section 8.12.1.51
0x48094	at_phyt_frame_tc_msbs		at_phyt_frame_tc_msbs	Section 8.12.1.52
0x48098	at_radt_init_lsbs		at_radt_init_lsbs	Section 8.12.1.53
0x4809C	at_radt_init_mid		at_radt_init_mid	Section 8.12.1.54
0x480A0	at_radt_init_msbs		at_radt_init_msbs	Section 8.12.1.55
0x480A4	at_ulradt_init_lsbs		at_ulradt_init_lsbs	Section 8.12.1.56
0x480A8	at_radt_tstamp_value		at_radt_tstamp_value	Section 8.12.1.57
0x480B0	at_dlradt_init_lsbs		at_dlradt_init_lsbs	Section 8.12.1.58
0x480B4	at_gsm_tcount_init		at_gsm_tcount initialization	Section 8.12.1.59
0x480B8	at_gsm_tcount_value		at_gsm_tcount value	Section 8.12.1.60
0x480BC	at_radt_symb_lut_index_tc		at_radt_symb_lut_index_tc	Section 8.12.1.61
0x480C8	at_radt_frame_tc_lsbs		at_radt_frame_tc_msbs	Section 8.12.1.62
0x480CC	at_radt_frame_tc_msbs		at_radt_frame_tc_lsbs	Section 8.12.1.63
0x480E0	at_ulradt_init_mid		at_ulradt_init_mid	Section 8.12.1.64
0x480E4	at_ulradt_init_msbs		at_ulradt_init_msbs	Section 8.12.1.65
0x480E8	at_dlradt_init_mid		at_dlradt_init_mid	Section 8.12.1.66
0x480EC	at_dlradt_init_msbs		at_dlradt_init_msbs	Section 8.12.1.67
0x480F4	at_radt_init_lut_index		at_radt LUT Index initial value	Section 8.12.1.68
0x480F8	at_ulradt_init_lut_index		at_ulradt LUT Index initial value	Section 8.12.1.69
0x480FC	at_dlradt_init_lut_index		at_dlradt LUT Index initial value	Section 8.12.1.70
0x48148	at_neg_delta		Negative delta control	Section 8.12.1.71
0x481F8	at_evt_enable		at_evt_enable	Section 8.12.1.72
0x481FC	at_evt_force		at_evt_force	Section 8.12.1.73

Table 8-195. AT Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x483F8	at_internal_evt_enable		at_internal_evt_enable. LSB for each field corresponds to the lowest event number for that field.	Section 8.12.1.74
0x483FC	at_internal_evt_force		at_internal_evt_force	Section 8.12.1.75
0x4A000	at_radt_sym_lut_ram		at_radt_sym_lut_ram	Section 8.12.1.76

8.12.1 Grouped Common Registers Details

8.12.1.1 at_pimax_lk[0] Register (offset = 0x48100)

Table 8-196. at_pimax_lk[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	pi_max	READ_WRITE	PI max window value per link. unit is byte clock
at_pimax_lk[1]	**Access = READ_WRITE**		**Address [0x4810C]**
at_pimax_lk[2]	**Access = READ_WRITE**		**Address [0x48118]**
at_pimax_lk[3]	**Access = READ_WRITE**		**Address [0x48124]**
at_pimax_lk[4]	**Access = READ_WRITE**		**Address [0x48130]**
at_pimax_lk[5]	**Access = READ_WRITE**		**Address [0x4813C]**

8.12.1.2 at_pimin_lk[0] Register (offset = 0x48104)

Table 8-197. at_pimin_lk[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	pi_min	READ_WRITE	PI min window value per link. unit is byte clock
at_pimin_lk[1]	**Access = READ_WRITE**		**Address [0x48110]**
at_pimin_lk[2]	**Access = READ_WRITE**		**Address [0x4811C]**
at_pimin_lk[3]	**Access = READ_WRITE**		**Address [0x48128]**
at_pimin_lk[4]	**Access = READ_WRITE**		**Address [0x48134]**
at_pimin_lk[5]	**Access = READ_WRITE**		**Address [0x48140]**

8.12.1.3 at_pivalue_lk[0] Register (offset = 0x48108)

Table 8-198. at_pivalue_lk[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	pi_capt	READ	PI captured value per link. unit is byte clock
at_pivalue_lk[1]	**Access = READ_WRITE**		**Address [0x48114]**
at_pivalue_lk[2]	**Access = READ_WRITE**		**Address [0x48120]**
at_pivalue_lk[3]	**Access = READ_WRITE**		**Address [0x4812C]**
at_pivalue_lk[4]	**Access = READ_WRITE**		**Address [0x48138]**
at_pivalue_lk[5]	**Access = READ_WRITE**		**Address [0x48144]**

8.12.1.4 at_event_offset[0] Register (offset = 0x48200)

Table 8-199. at_event_offset[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD1	N/A	RESERVED
26-24	evt_strb_sel	READ_WRITE	Strobe Selection 0 = Use RADT Symbol to trigger the event 1 = Use RADT Frame to trigger the event 2 = Use ULRADT Symbol to trigger the event 3 = Use ULRADT Frame to trigger the event 4 = Use DLRADT Symbol to trigger the event 5 = Use DLRADT Frame to trigger the event 6 = Use PHYT Frame to trigger the event
23-22	RSVD	N/A	RESERVED
21-0	evt_index	READ_WRITE	Event Offset Index. Should be smaller than evt_mod

at_event_offset[1]	**Access = READ_WRITE**	**Address [0x48210]**
at_event_offset[2]	**Access = READ_WRITE**	**Address [0x48220]**
at_event_offset[3]	**Access = READ_WRITE**	**Address [0x48230]**
at_event_offset[4]	**Access = READ_WRITE**	**Address [0x48240]**
at_event_offset[5]	**Access = READ_WRITE**	**Address [0x48250]**
at_event_offset[6]	**Access = READ_WRITE**	**Address [0x48260]**
at_event_offset[7]	**Access = READ_WRITE**	**Address [0x48270]**
at_event_offset[8]	**Access = READ_WRITE**	**Address [0x48280]**
at_event_offset[9]	**Access = READ_WRITE**	**Address [0x48290]**
at_event_offset[10]	**Access = READ_WRITE**	**Address [0x482A0]**

8.12.1.5 at_event_mod_tc[0] Register (offset = 0x48204)

Table 8-200. at_event_mod_tc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_mod	READ_WRITE	Set Event Modulo value minus one. Additional events are created every modulo time between strobe.

at_event_mod_tc[1]	**Access = READ_WRITE**	**Address [0x48214]**
at_event_mod_tc[2]	**Access = READ_WRITE**	**Address [0x48224]**
at_event_mod_tc[3]	**Access = READ_WRITE**	**Address [0x48234]**
at_event_mod_tc[4]	**Access = READ_WRITE**	**Address [0x48244]**
at_event_mod_tc[5]	**Access = READ_WRITE**	**Address [0x48254]**
at_event_mod_tc[6]	**Access = READ_WRITE**	**Address [0x48264]**
at_event_mod_tc[7]	**Access = READ_WRITE**	**Address [0x48274]**
at_event_mod_tc[8]	**Access = READ_WRITE**	**Address [0x48284]**
at_event_mod_tc[9]	**Access = READ_WRITE**	**Address [0x48294]**
at_event_mod_tc[10]	**Access = READ_WRITE**	**Address [0x482A4]**

8.12.1.6 at_event_mask_lsbs[0] Register (offset = 0x48204)

Table 8-201. at_event_mask_lsbs[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	evt_mask_l	READ_WRITE	Event Mask LSBs. Mainly used for GSM
at_event_mask_lsbs[1]	**Access = READ_WRITE**	**Address [0x48218]**	
at_event_mask_lsbs[2]	**Access = READ_WRITE**	**Address [0x48228]**	
at_event_mask_lsbs[3]	**Access = READ_WRITE**	**Address [0x48238]**	
at_event_mask_lsbs[4]	**Access = READ_WRITE**	**Address [0x48248]**	
at_event_mask_lsbs[5]	**Access = READ_WRITE**	**Address [0x48258]**	
at_event_mask_lsbs[6]	**Access = READ_WRITE**	**Address [0x48268]**	
at_event_mask_lsbs[7]	**Access = READ_WRITE**	**Address [0x48278]**	
at_event_mask_lsbs[8]	**Access = READ_WRITE**	**Address [0x48288]**	
at_event_mask_lsbs[9]	**Access = READ_WRITE**	**Address [0x48298]**	
at_event_mask_lsbs[10]	**Access = READ_WRITE**	**Address [0x482A8]**	

8.12.1.7 at_event_mask_msbs[0] Register (offset = 0x4820C)

Table 8-202. at_event_mask_msbs[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-0	evt_mask_h	READ_WRITE	Event Mask MSBs. Mainly used for GSM
at_event_mask_msbs[1]		**Access = READ_WRITE**	**Address [0x4821C]**
at_event_mask_msbs[2]		**Access = READ_WRITE**	**Address [0x4822C]**
at_event_mask_msbs[3]		**Access = READ_WRITE**	**Address [0x4823C]**
at_event_mask_msbs[4]		**Access = READ_WRITE**	**Address [0x4824C]**
at_event_mask_msbs[5]		**Access = READ_WRITE**	**Address [0x4825C]**
at_event_mask_msbs[6]		**Access = READ_WRITE**	**Address [0x4826C]**
at_event_mask_msbs[7]		**Access = READ_WRITE**	**Address [0x4827C]**
at_event_mask_msbs[8]		**Access = READ_WRITE**	**Address [0x4828C]**
at_event_mask_msbs[9]		**Access = READ_WRITE**	**Address [0x4829C]**
at_event_mask_msbs[10]		**Access = READ_WRITE**	**Address [0x482AC]**

8.12.1.8 at_ad_ingr_event_offset[0] Register (offset = 0x48400)

Table 8-203. at_ad_ingr_event_offset[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD1	N/A	RESERVED
26-24	evt_strb_sel	READ_WRITE	Strobe Select 0 = Use RADT Symbol to trigger the event 1 = Use RADT Frame to trigger the event 2 = Use ULRADT Symbol to trigger the event 3 = Use ULRADT Frame to trigger the event 4 = Use DLRADT Symbol to trigger the event 5 = Use DLRADT Frame to trigger the event
23-22	RSVD	N/A	RESERVED
21-0	evt_index	READ_WRITE	Event Offset Index. at_ad_ingr_event_offset [0] ~ [2] is used for WCDMA chip event for each DIO engine and at_ad_ingr_event_offset [3] ~ [5] is used for Frame event for each DIO engine

at_ad_ingr_event_offset[1]	**Access = READ_WRITE**	**Address [0x48408]**
at_ad_ingr_event_offset[2]	**Access = READ_WRITE**	**Address [0x48410]**
at_ad_ingr_event_offset[3]	**Access = READ_WRITE**	**Address [0x48418]**
at_ad_ingr_event_offset[4]	**Access = READ_WRITE**	**Address [0x48420]**
at_ad_ingr_event_offset[5]	**Access = READ_WRITE**	**Address [0x48428]**

8.12.1.9 at_ad_ingr_event_mod_tc[0] Register (offset = 0x48404)

Table 8-204. at_ad_ingr_event_mod_tc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_mod	READ_WRITE	Set Event Modulo value minus one. Additional events are created every modulo time between strobe. at_ad_ingr_event_mod_tc[0] ~ [2] is used for WCDMA chip event for each DIO engine and at_ad_ingr_event_mod_tc[3] ~ [5] is used for Frame event for each DIO engine

at_ad_ingr_event_mod_tc[1]	**Access = READ_WRITE**	**Address [0x4840C]**
at_ad_ingr_event_mod_tc[2]	**Access = READ_WRITE**	**Address [0x48414]**
at_ad_ingr_event_mod_tc[3]	**Access = READ_WRITE**	**Address [0x4841C]**
at_ad_ingr_event_mod_tc[4]	**Access = READ_WRITE**	**Address [0x48424]**
at_ad_ingr_event_mod_tc[5]	**Access = READ_WRITE**	**Address [0x4842C]**

8.12.1.10 at_ad_egr_event_offset[0] Register (offset = 0x48500)

Table 8-205. at_ad_egr_event_offset[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD1	N/A	RESERVED
26-24	evt_strb_sel	READ_WRITE	Strobe Select 0 = Use RADT Symbol to trigger the event 1 = Use RADT Frame to trigger the event 2 = Use ULRADT Symbol to trigger the event 3 = Use ULRADT Frame to trigger the event 4 = Use DLRADT Symbol to trigger the event 5 = Use DLRADT Frame to trigger the event
23-22	RSVD	N/A	RESERVED
21-0	evt_index	READ_WRITE	Event Offset Index. at_ad_egr_event_offset [0] ~ [2] is used for WCDMA chip event for each DIO engine and at_ad_egr_event_offset [3] ~ [5] is used for Frame event for each DIO engine

at_ad_egr_event_offset[1]	**Access = READ_WRITE**	**Address [0x48508]**
at_ad_egr_event_offset[2]	**Access = READ_WRITE**	**Address [0x48510]**
at_ad_egr_event_offset[3]	**Access = READ_WRITE**	**Address [0x48518]**
at_ad_egr_event_offset[4]	**Access = READ_WRITE**	**Address [0x48520]**
at_ad_egr_event_offset[5]	**Access = READ_WRITE**	**Address [0x48528]**

8.12.1.11 at_ad_egr_event_mod_tc[0] Register (offset = 0x48504)

Table 8-206. at_ad_egr_event_mod_tc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_mod	READ_WRITE	Set Event Modulo value minus one. Additional events are created every modulo time between strobe. at_ad_egr_event_mod_tc[0] ~ [2] is used for WCDMA chip event for each DIO engine and at_ad_egr_event_mod_tc[3] ~ [5] is used for Frame event for each DIO engine

at_ad_egr_event_mod_tc[1]	**Access = READ_WRITE**	**Address [0x4850C]**
at_ad_egr_event_mod_tc[2]	**Access = READ_WRITE**	**Address [0x48514]**
at_ad_egr_event_mod_tc[3]	**Access = READ_WRITE**	**Address [0x4851C]**
at_ad_egr_event_mod_tc[4]	**Access = READ_WRITE**	**Address [0x48524]**
at_ad_egr_event_mod_tc[5]	**Access = READ_WRITE**	**Address [0x4852C]**

8.12.1.12 at_tm_delta_event_offset[0] Register (offset = 0x48540)

Table 8-207. at_tm_delta_event_offset[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_index	READ_WRITE	Delta Offset per link. unit is byte clock
at_tm_delta_event_offset[1]		**Access = READ_WRITE**	**Address [0x48548]**
at_tm_delta_event_offset[2]		**Access = READ_WRITE**	**Address [0x48550]**
at_tm_delta_event_offset[3]		**Access = READ_WRITE**	**Address [0x48558]**
at_tm_delta_event_offset[4]		**Access = READ_WRITE**	**Address [0x48560]**
at_tm_delta_event_offset[5]		**Access = READ_WRITE**	**Address [0x48568]**

8.12.1.13 at_tm_delta_event_mod_tc[0] Register (offset = 0x48544)

Table 8-208. at_tm_delta_event_mod_tc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_mod	READ_WRITE	Event Modulo. Normally set to one frame length minus one.CSL sets this value automatically
at_tm_delta_event_mod_tc[1]	**Access = READ_WRITE**	**Address [0x4854C]**	
at_tm_delta_event_mod_tc[2]	**Access = READ_WRITE**	**Address [0x48554]**	
at_tm_delta_event_mod_tc[3]	**Access = READ_WRITE**	**Address [0x4855C]**	
at_tm_delta_event_mod_tc[4]	**Access = READ_WRITE**	**Address [0x48564]**	
at_tm_delta_event_mod_tc[5]	**Access = READ_WRITE**	**Address [0x4856C]**	

8.12.1.14 at_pe_event_offset[0] Register (offset = 0x48580)

Table 8-209. at_pe_event_offset[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_index	READ_WRITE	PE1 Event Offset Index per link. unit is byte clock It allows PE to prepare data insertion and aggregation on RT

at_pe_event_offset[1]	**Access = READ_WRITE**	**Address [0x48588]**
at_pe_event_offset[2]	**Access = READ_WRITE**	**Address [0x48590]**
at_pe_event_offset[3]	**Access = READ_WRITE**	**Address [0x48598]**
at_pe_event_offset[4]	**Access = READ_WRITE**	**Address [0x485A0]**
at_pe_event_offset[5]	**Access = READ_WRITE**	**Address [0x485A8]**

8.12.1.15 at_pe_event_mod_tc[0] Register (offset = 0x48584)

Table 8-210. at_pe_event_mod_tc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_mod	READ_WRITE	PE1 Event Modulo. Normally set to one frame length minus one. CSL sets this value automatically
at_pe_event_mod_tc[1]		**Access = READ_WRITE**	**Address [0x4858C]**
at_pe_event_mod_tc[2]		**Access = READ_WRITE**	**Address [0x48594]**
at_pe_event_mod_tc[3]		**Access = READ_WRITE**	**Address [0x4859C]**
at_pe_event_mod_tc[4]		**Access = READ_WRITE**	**Address [0x485A4]**
at_pe_event_mod_tc[5]		**Access = READ_WRITE**	**Address [0x485AC]**

8.12.1.16 at_pe_event2_offset[0] Register (offset = 0x485B0)

Table 8-211. at_pe_event2_offset[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_index	READ_WRITE	PE2 Event Offset Index per link. unit is byte clock. It allows PE to prepare frame data insertion and channel enable.
at_pe_event2_offset[1]		**Access = READ_WRITE**	**Address [0x485B8]**
at_pe_event2_offset[2]		**Access = READ_WRITE**	**Address [0x485C0]**
at_pe_event2_offset[3]		**Access = READ_WRITE**	**Address [0x485C8]**
at_pe_event2_offset[4]		**Access = READ_WRITE**	**Address [0x485D0]**
at_pe_event2_offset[5]		**Access = READ_WRITE**	**Address [0x485D8]**

8.12.1.17 at_pe_event2_mod_tc[0] Register (offset = 0x485B4)

Table 8-212. at_pe_event2_mod_tc[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	evt_mod	READ_WRITE	PE 2 Event Modulo. Normally one frame length minus one. CSL sets this value automatically
at_pe_event2_mod_tc[1]	**Access = READ_WRITE**	**Address [0x485BC]**	
at_pe_event2_mod_tc[2]	**Access = READ_WRITE**	**Address [0x485C4]**	
at_pe_event2_mod_tc[3]	**Access = READ_WRITE**	**Address [0x485CC]**	
at_pe_event2_mod_tc[4]	**Access = READ_WRITE**	**Address [0x485D4]**	
at_pe_event2_mod_tc[5]	**Access = READ_WRITE**	**Address [0x485DC]**	

8.12.1.18 at_control1 Register (offset = 0x48000)

Table 8-213. at_control1 Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD	N/A	RESERVED
28-23	rp1_phyt_fr_ld	READ_WRITE	RP1 PHYT Frame Load Size. size +8 value will be set. Min size is 0 (8bits) and Max size is 32 (40 bits)
22-17	rp1_radt_fr_ld	READ_WRITE	RP1 RADT Frame Load Size. size +8 value will be set Min size is 0 (8bits) and Max size is 32 (40 bits)
16-13	sync_sampl_window	READ_WRITE	+ or - sync_sampl_window value = size. Full window is ± size, min size is ± 1
12	crc_invert	READ_WRITE	CRC invert 0 = No CRC data invert 1 = CRC data Invert
11	crc_init_ones	READ_WRITE	CRC init ones 0 = No init. CRC calculator is initialized to 0x0000 1 = Init ones. CRC calculator is initialized to 0xFFFF
10	crc_flip	READ_WRITE	CRC flip selection 0 = Do not flip. MSB of the CRC data is sent first 1 = CRC flip. LSB of the CRC data is sent first
9	crc_use	READ_WRITE	CRC Mode 0 = Do not use CRC 1 = CRC use
8	auto_resync	READ_WRITE	Auto resync if new frame boundary is received 0 = Non Auto 1 = Auto resync
7	rp1_mode	READ_WRITE	RP1 mode 0 = Non – RP1 mode 1 = RP1 mode
6-3	rad_syncsel	READ_WRITE	RAD sync selection option 0 = Use RP1 interface for synchronizing the RADT frame boundary 1 = Use RADTSYNC chip input for synchronizing the RADT frame boundary 2 = Use software mmr at_sw_sync for synchronizing the RADT frame boundary 3 = Use Received frame boundary for synchronizing the RADT frame boundary 4 = Use compared PHYT value for synchronizing the RADT frame boundary
2-0	phy_syncsel	READ_WRITE	PHY sync selection option 0 = Use RP1 interface for synchronizing the PHYT frame boundary 1 = Use PHYTSYNC chip input for synchronizing the PHYT frame boundary 2 = Use software mmr at_sw_sync for synchronizing the PHYT frame boundary 3 = Use Received frame boundary for synchronizing the PHYT frame boundary

8.12.1.19 at_control2 Register (offset = 0x48004)

Table 8-214. at_control2 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	N/A	RESERVED
1	arm_timer	READ_WRITE	Arm Timer 0 = Do not arm 1 = arm
0	halt_timer	READ_WRITE	Halt Timer 0 = Do not halt 1 = Halt

8.12.1.20 at_sw_sync Register (offset = 0x48008)

Table 8-215. at_sw_sync Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	N/A	RESERVED
1	phyt_sync	WRITE	PHY sync 1 = Generate Trigger
0	radt_sync	WRITE	RAD sync 1 = Generate Trigger

8.12.1.21 at_phyt_cmp_radsync Register (ofset = 0x4800C)

Table 8-216. at_phyt_cmp_radsync Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	phyt_cmp	READ_WRITE	PHYT_CMP PHYT compares value to produce RADT sync when selected. If this value is zero, it means RADT and PHYT starts almost at the same time (there could be one or two clock delay from the HW)

8.12.1.22 at_rp1_type Register (offset = 0x48010)

Table 8-217. at_rp1_type Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	N/A	RESERVED
7-0	rp1_radt_type_sel	READ_WRITE	RP1 RAD Type Selection

8.12.1.23 at_captradt Register (offset = 0x48014)

Table 8-218. at_captradt Register Field Descriptions

Bits	Field Name	Type	Description
31-27	radt_fr_cap	READ	Capture RADT frame count 5lsbs upon a PHYT frame boundary
26-19	radt_symb_cap	READ	Capture RADT symbol count upon a PHYT frame boundary
18-0	radt_clk_cap	READ	Capture RADT clock count upon a PHYT frame boundary

8.12.1.24 at_rp1_type_capture Register (offset = 0x48020)

Table 8-219. at_rp1_type_capture Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	N/A	RESERVED
7-0	rp1_type_capt	READ	RP1 Type Captured

8.12.1.25 at_rp1_tod_capture_l Register (offset = 0x48024)

Table 8-220. at_rp1_tod_capture_l Register Field Descriptions

Bits	Field Name	Type	Description
31-0	rp1_tod_cap_l	READ	RP1 TOD Capture LSBs

8.12.1.26 at_rp1_tod_capture_h Register (offset = 0x48028)

Table 8-221. at_rp1_tod_capture_h Register Field Descriptions

Bits	Field Name	Type	Description
31-0	rp1_tod_cap_h	READ	RP1 TOD Capture MSBs

8.12.1.27 at_rp1_rp3_capture_l Register (offset = 0x4802C)

Table 8-222. at_rp1_rp3_capture_l Register Field Descriptions

Bits	Field Name	Type	Description
31-0	rp1_rp3_cap_l	READ	RP1 RP3 Capture LSBs

8.12.1.28 at_rp1_rp3_capture_h Register (offset = 0x48030)

Table 8-223. at_rp1_rp3_capture_h Register Field Descriptions

Bits	Field Name	Type	Description
31-0	rp1_rp3_cap_h	READ	RP1 RP3 Capture MSBs

8.12.1.29 at_rp1_rad_capture_l Register (offset = 0x48034)

Table 8-224. at_rp1_rad_capture_l Register Field Descriptions

Bits	Field Name	Type	Description
31-0	rp1_rad_cap_l	READ	RP1 RADIO System Capture LSBs

8.12.1.30 at_rp1_rad_capture_h Register (offset = 0x48038)

Table 8-225. at_rp1_rad_capture_h Register Field Descriptions

Bits	Field Name	Type	Description
31-0	rp1_rad_cap_h	READ	RP1 RADIO System Capture MSBs

8.12.1.31 at_phyt_clkcnt_value Register (offset = 0x48040)

Table 8-226. at_phyt_clkcnt_value Register Field Descriptions

Bits	Field Name	Type	Description
31-24	RSVD	N/A	RESERVED
23-0	phy_clkcnt_val	READ	PHYT clock count Value, 2 lsbs are always 00

8.12.1.32 at_phyt_frm_value_lsbs Register (offset = 0x48044)

Table 8-227. at_phyt_frm_value_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	phyt_fr_val_l	READ	PHYT Frame Value LSBs

8.12.1.33 at_phyt_frm_value_msbs Register (offset = 0x48048)

Table 8-228. at_phyt_frm_value_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	phyt_fr_val_h	READ	PHYT Frame Value MSBs

8.12.1.34 at_radt_value_lsbs Register (offset = 0x4804C)

Table 8-229. at_radt_value_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD	N/A	RESERVED
26-19	radt_symcnt_val	READ	RADT symbol count Value
18-0	radt_clkcnt_val	READ	RADT clock count Value

8.12.1.35 t_radt_value_mid Register (offset = 0x48050)

Table 8-230. at_radt_value_mid Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	radt_fr_val_l	READ	RADT Frame Value LSBs

8.12.1.36 at_radt_value_msbs Register (offset = 0x48054)

Table 8-231. at_radt_value_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	radt_fr_val_h	READ	RADT Frame Value MSBs

8.12.1.37 at_ulradt_value_lsbs Register (offset = 0x48058)

Table 8-232. at_ulradt_value_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD	N/A	RESERVED
26-19	ulradt_symcnt_val	READ	ULRADT symbol count Values
18-0	ulradt_clkcnt_val	READ	ULRADT clock count Value

8.12.1.38 at_ulradt_value_mid Register (offset = 0x4805C)

Table 8-233. at_ulradt_value_mid Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	ulradt_fr_val_l	READ	ULRADT Frame Value LSBs

8.12.1.39 at_ulradt_value_msbs Register (offset = 0x48060)

Table 8-234. at_ulradt_value_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	ulradt_fr_val_h	READ	ULRADT Frame Value MSBs

8.12.1.40 at_dlradt_value_lsbs Register (offset = 0x48064)

Table 8-235. at_dlradt_value_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD	N/A	RESERVED
26-19	dlradt_symcnt_val	READ	DLRADT symbol count Value
18-0	dlradt_clkcnt_val	READ	DLRADT clock count Value

8.12.1.41 at_dlradt_value_mid Register (offset = 0x48068)

Table 8-236. at_dlradt_value_mid Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	dlradt_fr_val_l	READ	DLRADT Frame Value LSBs

8.12.1.42 at_dlradt_value_msbs Register (offset = 0x4806C)

Table 8-237. at_dlradt_value_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	dlradt_fr_val_h	READ	DLRADT Frame Value MSBs

8.12.1.43 at_radt_wcdma_value Register (offset = 0x48070)

Table 8-238. at_radt_wcdma_value Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-16	radt_fr	READ	Frame Value
15-12	radt_slot	READ	Slot Value
11-0	radt_chip	READ	Chip Value

8.12.1.44 at_ulradt_wcdma_value Register (offset = 0x48074)

Table 8-239. at_ulradt_wcdma_value Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-16	ulradt_fr	READ	Frame Value
15-12	ulradt_slot	READ	Slot Value
11-0	ulradt_chip	READ	Chip Value

8.12.1.45 at_dlradt_wcdma_value Register (offset = 0x48078)

Table 8-240. at_dlradt_wcdma_value Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-16	dlradt_fr	READ	Frame Value
15-12	dlradt_slot	READ	Slot Value
11-0	dlradt_chip	READ	Chip Value

8.12.1.46 at_radt_wcdma_div Register (offset = 0x4807C)

Table 8-241. at_radt_wcdma_div Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	N/A	RESERVED
6-0	tc	READ_WRITE	at radt wcdma clock divider terminal count. This counter divides dualbyte clock down to 3.84MHz chip rate for wcdma counters. Normally it is set to 79 for OBSAI and 63 for CPRI

8.12.1.47 at_phyt_init_lsbs Register (offset = 0x48080)

Table 8-242. at_phyt_init_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	phyt_clkcnt_init	READ_WRITE	PHYT clock count Init

8.12.1.48 at_phyt_init_mid Register (offset = 0x48084)

Table 8-243. at_phyt_init_mid Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	phyt_fr_init_l	READ_WRITE	PHYT Frame Init LSBs

8.12.1.49 at_phyt_init_msbs Register (offset = 0x48088)

Table 8-244. at_phyt_init_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	phyt_fr_init_h	READ_WRITE	PHYT Frame Init MSBs

8.12.1.50 at_phyt_tc_lsbs Register (offset = 0x4808C)

Table 8-245. at_phyt_tc_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-0	phyt_clkcnt_tc	READ_WRITE	PHYT Clock Counter TC

8.12.1.51 at_phyt_frame_tc_lsbs Register (offset = 0x48090)

Table 8-246. at_phyt_frame_tc_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-0	phyt_fr_tc_l	READ_WRITE	PHYT Frame TC LSBs

8.12.1.52 at_phyt_frame_tc_msbs Register (offset = 0x48094)

Table 8-247. at_phyt_frame_tc_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD		RESERVED
7-0	phyt_fr_tc_h	READ_WRITE	PHYT Frame TC MSBs

8.12.1.53 at_radt_init_lsbs Register (offset = 048098)

Table 8-248. at_radt_init_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD	N/A	RESERVED
26-19	radt_symcnt_init	READ_WRITE	RADT symbol count Init
18-0	radt_clkcnt_init	READ_WRITE	RADT clock count Init

8.12.1.54 at_radt_init_mid Register (offset = 0x4809C)

Table 8-249. at_radt_init_mid Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	radt_fr_init_l	READ_WRITE	RADT Frame Init LSBs

8.12.1.55 at_radt_init_msbs Register (offset = 0x480A0)

Table 8-250. at_radt_init_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	radt_fr_init_h	READ_WRITE	RADT Frame Init MSBs

8.12.1.56 at_ulradt_init_lsbs Register (offset = 0x480A4)

Table 8-251. at_ulradt_init_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31	ulfcp_minusone	READ_WRITE	ULFCB_minusone 0 = Do not minus one 1 = Minus one. Used when clock and symbol init timing is ahead of Rad timer. The fcb minus one only works when the frame number is captured by the RP1 interface. Otherwise, it must be initialized by software with the adjusted UL frame count
30-28	RSVD	N/A	RESERVED
26-19	ulradt_symcnt_init	READ_WRITE	ULRADT symbol count Init
18-0	ulradt_clkcnt_init	READ_WRITE	ULRADT clock count Init

8.12.1.57 at_radt_tstamp_value Register (offset = 0x480A8)

Table 8-252. at_radt_tstamp_value Register Field Descriptions

Bits	Field Name	Type	Description
31-25	RSVD	N/A	RESERVED
24-0	radt_tstamp	READ	RADT timestamp clock counter value. This shows capture of the offset of where we are in time within a radio frame. Max number is the radio frame length in dual byte clock.

8.12.1.58 at_dlradt_init_lsbs Register (offset = 0x480B0)

Table 8-253. at_dlradt_init_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31	dlfcb_minusone	READ_WRITE	DLFCB_minusone 0 = No activity 1 = Minus one. Used when clock and symbol init timing is ahead of Rad timer. The fcb minus one only works when the frame number is captured by the RP1 interface. Otherwise, it must be initialized by software with the adjusted UL frame count
30-28	RSVD	N/A	RESERVED
26-19	dlradt_symcnt_init	READ_WRITE	DLRADT symbol count Init
18-0	dlradt_clkcnt_init	READ_WRITE	DLRADT clock count Init

8.12.1.59 at_gsm_tcount_init Register (offset = 0x480B4)

Table 8-254. at_gsm_tcount_init Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-16	t3	READ_WRITE	T3 w=init, affects at_gsm_tcount_value
15-11	t2	READ_WRITE	T2 w=init, affects at_gsm_tcount_value
10-0	t1	READ_WRITE	T1 w=init, affects at_gsm_tcount_value

8.12.1.60 at_gsm_tcount_value Register (offset = 0x480D8)

Table 8-255. at_gsm_tcount_value Register Field Descriptions

Bits	Field Name	Type	Description
31-22	RSVD	N/A	RESERVED
21-16	t3	READ	T3 r=value of T3 counter
15-11	t2	READ	T2 r=value of T2 counter
10-0	t1	READ	T1 r=value of T1 counter

8.12.1.61 at_radt_symb_lut_index_tc Register (offset = 0x480BC)

Table 8-256. at_radt_symb_lut_index_tc Register

Bits	Field Name	Type	Description
31-16	RSVD1	N/A	RESERVED
15-8	radt_sym_tc	READ_WRITE	Symbol TC
7	RSVD	N/A	RESERVED
6-0	radt_lutindex_tc	READ_WRITE	LUT Index TC

8.12.1.62 at_radt_frame_tc_lsbs Register (offset = 0x480C8)

Table 8-257. at_radt_frame_tc_lsbs Register Field Descriptions

Bits	Field Name	Type	Description
31-0	radt_fr_tc_l	READ_WRITE	RADT Frame TC LSBs

8.12.1.63 at_radt_frame_tc_msbs Register (offset = 0x480CC)

Table 8-258. at_radt_frame_tc_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	N/A	RESERVED
7-0	radt_fr_tc_h	READ_WRITE	RADT Frame TC MSBs

8.12.1.64 at_ulradt_init_mid Register (offset = 0x480E0)

Table 8-259. at_ulradt_init_mid Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	ulradt_fr_init_l	READ_WRITE	RADT Frame Init LSBs

8.12.1.65 at_ulradt_init_msbs Register (offset = 0x480E4)

Table 8-260. at_ulradt_init_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	ulradt_fr_init_h	READ_WRITE	RADT Frame Init MSBs

8.12.1.66 at_dlradt_init_mid Register (offset = 0x480E8)

Table 8-261. at_dlradt_init_mid Register Field Descriptions

Bits	Field Name	Type	Description
31-12	RSVD	N/A	RESERVED
11-0	dlradt_fr_init_l	READ_WRITE	RADT Frame Init LSBs

8.12.1.67 at_dlradt_init_msbs Register (offset = 0x480EC)

Table 8-262. at_dlradt_init_msbs Register Field Descriptions

Bits	Field Name	Type	Description
31-28	RSVD	N/A	RESERVED
27-0	dlradt_fr_init_h	READ_WRITE	RADT Frame Init MSBs

8.12.1.68 at_radt_init_lut_index Register (offset = 0x480F4)

Table 8-263. at_radt_init_lut_index Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	N/A	RESERVED
6-0	radt_lutindex_init	READ_WRITE	LUT Index initial value

8.12.1.69 at_ulradt_init_lut_index Register (offset = 0x480F8)

Table 8-264. at_ulradt_init_lut_index Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	N/A	RESERVED
6-0	ulradt_lutindex_init	READ_WRITE	LUT Index initial value

8.12.1.70 at_dlradt_init_lut_index Register (offset = 0x480FC)

Table 8-265. at_dlradt_init_lut_index Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	N/A	RESERVED
6-0	dlradt_lutindex_init	READ_WRITE	LUT Index initial value

8.12.1.71 at_neg_delta Register (offset = 0x48148)

Table 8-266. at_neg_delta Register Field Descriptions

Bits	Field Name	Type	Description
31-6	RSVD	N/A	RESERVED
5	lk5_delta	READ_WRITE	Control for positive or negative delta for TM delta event. Controls the BFN PHYT frame number passed to the TM when the Delta event occurs 0 = TM delta event is positive BFN = PHYT frame num 1 = TM delta event is positive BFN = PHYT frame num minus 1
4	lk4_delta	READ_WRITE	Control for positive or negative delta for TM delta event. Controls the BFN PHYT frame number passed to the TM when the Delta event occurs 0 = TM delta event is positive BFN = PHYT frame num 1 = TM delta event is positive BFN = PHYT frame num minus 1
3	lk3_delta	READ_WRITE	Control for positive or negative delta for TM delta event. Controls the BFN PHYT frame number passed to the TM when the Delta event occurs 0 = TM delta event is positive BFN = PHYT frame num 1 = TM delta event is positive BFN = PHYT frame num minus 1
2	lk2_delta	READ_WRITE	Control for positive or negative delta for TM delta event. Controls the BFN PHYT frame number passed to the TM when the Delta event occurs 0 = TM delta event is positive BFN = PHYT frame num 1 = TM delta event is positive BFN = PHYT frame num minus 1
1	lk1_delta	READ_WRITE	Control for positive or negative delta for TM delta event. Controls the BFN PHYT frame number passed to the TM when the Delta event occurs 0 TM delta event is positive BFN = PHYT frame num 1 TM delta event is positive BFN = PHYT frame num minus 1
0	lk0_delta	READ_WRITE	Control for positive or negative delta for TM delta event. Controls the BFN PHYT frame number passed to the TM when the Delta event occurs 0 = TM delta event is positive BFN = PHYT frame num 1 = TM delta event is positive BFN = PHYT frame num minus 1

8.12.1.72 at_evt_enable Register (offset = 0x481F8)

Table 8-267. at_evt_enable Register Field Descriptions

Bits	Field Name	Type	Description
31-11	RSVD	N/A	RESERVED
10-0	evt_en	READ_WRITE	EVENT Enable for 11 external events bit0:Event0 ~ bit 10: Event10

8.12.1.73 at_evt_force Register (offset = 0x481FC)

Table 8-268. at_evt_force Register Field Descriptions

Bits	Field Name	Type	Description
31-11	RSVD	N/A	RESERVED
10-0	evt_force	WRITE	EVENT Force for 11 external events bit0:Event0 ~ bit10:Event10

8.12.1.74 at_internal_evt_enable Register (offset = 0x483F8)

Table 8-269. at_internal_evt_enable Register Field Descriptions

Bits	Field Name	Type	Description
31-30	RSVD	N/A	RESERVED
29-24	pe_evt2_en	READ_WRITE	PE2 EVENT Enable. One event per link. LSB goes to link0, MSB to link5
23-18	pe_evt_en	READ_WRITE	PE 1 EVENT Enable. One event per link. LSB goes to link0, MSB to link5
17-12	tm_delta_evt_en	READ_WRITE	TM DELTA EVENT Enable. One event per link. LSB goes to link0, MSB to link5
11-6	ad_egr_evt_en	READ_WRITE	AD Egress EVENT Enable. 3 LSB bits go to egress DIO DMA events for three engines and 3 MSB bits go to egress Frame event for three engines
5-0	ad_igr_evt_en	READ_WRITE	AD Ingress EVENT Enable. 3 LSB bits go to ingress DIO DMA events for three engines and 3 MSB bits go to ingress Frame event for three engines

8.12.1.75 at_internal_evt_force Register (offset = 0x483FC)

Table 8-270. at_internal_evt_force Register Field Descriptions

Bits	Field Name	Type	Description
31-30	RSVD	N/A	RESERVED
29-24	pe_evt2_force	WRITE	PE 2 EVENT Force
23-18	pe_evt_force	WRITE	PE1 EVENT Force
17-12	tm_delta_evt_force	WRITE	TM DELTA EVENT Force
11-6	ad_egr_evt_force	WRITE	AD Egress EVENT Force
5-0	ad_igr_evt_force	WRITE	AD Ingress EVENT Force

8.12.1.76 at_radt_sym_lut_ram[128] Register (offset = 0x4A000)

Table 8-271. at_radt_sym_lut_ram[128] Register Field Descriptions

Bits	Field Name	Type	Description
18-0	radt_clkcnt_tc	READ_WRITE	Radt clock count TC. It is also used for Ul and Dl
31-19	RSVD	N/A	RESERVED

8.13 EE Registers

Table 8-272. EE Register Memory Map

Offset	Register Name	Access	Description	Section
0x4000	EE_VB_EOI		This register supports the End of Interrupt (EOI) interface between the AIF2 and the external Interrupt Distributor Component used in the Highlander 0.1 Interrupt Architecture. It is a common register that is used for servicing both of the AIF2 interrupts. This register is written by software to acknowledge the interrupt has been cleared so another interrupt of the same type can be generated by the Interrupt Distributor Component. This register is written after a AIF2 interrupt has been cleared via the associated Interrupt Clear register. Writes to this register have no effect on the internal interrupt processing of the AIF2.	Section 8.13.1.1
0x4004	EE_VB_INTR_SET		This register allows software to create an AIF2 Error Interrupt or AIF2 Alarm Interrupt by writing bits in this register. This register is provided for debug and diagnostics.	Section 8.13.1.2
0x4008	EE_VB_INTR_CLR		This register allows software to clear an AIF2 Error Interrupt or AIF2 Alarm Interrupt by writing bits in this register. This register is provided for debug and diagnostics.	Section 8.13.1.3
0x4100	ee_db_irs		EE DB Raw Status Register	Section 8.13.1.4
0x4104	ee_db_irs_set		EE DB Set Status Register	Section 8.13.1.5
0x4108	ee_db_irs_clr		EE DB Clear Status Register	Section 8.13.1.6
0x410C	ee_db_en_ev0		EE DB Enable EV0 Register	Section 8.13.1.7
0x4110	ee_db_en_set_ev0		EE DB Set Enable EV0 Register	Section 8.13.1.8
0x4114	ee_db_en_clr_ev0		EE DB Clear Enable EV0 Register	Section 8.13.1.9
0x4118	ee_db_en_ev1		EE DB Enable EV1 Register	Section 8.13.1.10
0x411C	ee_db_en_set_ev1		EE DB Set Enable EV1 Register	Section 8.13.1.11
0x4120	ee_db_en_clr_ev1		EE DB Clear Enable EV1 Register	Section 8.13.1.12
0x4124	ee_db_en_sts_ev0		EE DB Enabled Status EV0 Register	Section 8.13.1.13
0x4128	ee_db_en_sts_ev1		EE DB Enabled Status EV1 Register	Section 8.13.1.14
0x4200	ee_ad_irs		EE AD Raw Status Register	Section 8.13.1.15
0x4204	ee_ad_irs_set		EE AD Set Status Register	Section 8.13.1.16
0x4208	ee_ad_irs_clr		EE AD Clear Status Register	Section 8.13.1.17
0x420C	ee_ad_en_ev0		EE AD Enable EV0 Register	Section 8.13.1.18
0x4210	ee_ad_en_set_ev0		EE AD Set Enable EV0 Register	Section 8.13.1.19
0x4214	ee_ad_en_clr_ev0		EE AD Clear Enable EV0 Register	Section 8.13.1.20
0x4218	ee_ad_en_ev1		EE AD Enable EV1 Register	Section 8.13.1.21
0x421C	ee_ad_en_set_ev1		EE AD Set Enable EV1 Register	Section 8.13.1.22
0x4220	ee_ad_en_clr_ev1		EE AD Clear Enable EV1 Register	Section 8.13.1.23
0x4224	ee_ad_en_sts_ev0		EE AD Enabled Status EV0 Register	Section 8.13.1.24
0x4228	ee_ad_en_sts_ev1		EE AD Enabled Status EV1 Register	Section 8.13.1.25
0x4300	ee_cd_irs		EE CD Raw Status Register	Section 8.13.1.26
0x4304	ee_cd_irs_set		EE CD Set Status Register	Section 8.13.1.27
0x4308	ee_cd_irs_clr		EE CD Clear Status Register	Section 8.13.1.28
0x430C	ee_cd_en_ev		EE CD Enable EV0 Register	Section 8.13.1.29
0x4310	ee_cd_en_set_ev		EE CD Set Enable EV0 Register	Section 8.13.1.30
0x4314	ee_cd_en_clr_ev		EE CD Clear Enable EV0 Register	Section 8.13.1.31
0x4318	ee_cd_en_sts_ev		EE CD Enabled Status EV0 Register	Section 8.13.1.32

Table 8-272. EE Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x4400	ee_sd_irs		EE SD Raw Status Register	Section 8.13.1.33
0x4404	ee_sd_irs_set		EE SD Set Status Register	Section 8.13.1.34
0x4408	ee_sd_irs_clr		EE SD Clear Status Register	Section 8.13.1.35
0x440C	ee_sd_en_ev0		EE SD Enable EV0 Register	Section 8.13.1.36
0x4410	ee_sd_en_set_ev0		EE SD Enable Set EV0 Register	Section 8.13.1.37
0x4414	ee_sd_en_clr_ev0		EE SD Enable Clear EV0 Register	Section 8.13.1.38
0x4418	ee_sd_en_ev1		EE SD Enable EV1 Register	Section 8.13.1.39
0x441C	ee_sd_en_set_ev1		EE SD Enable Set EV1 Register	Section 8.13.1.40
0x4420	ee_sd_en_clr_ev1		EE SD Enable Clear EV1 Register	Section 8.13.1.41
0x4424	ee_sd_en_sts_ev0		EE SD Enabled Status EV0 Register	Section 8.13.1.42
0x4428	ee_sd_en_sts_ev1		EE SD Enabled Status EV1 Register	Section 8.13.1.43
0x4500	ee_vc_irs		EE VC Raw Status Register	Section 8.13.1.44
0x4504	ee_vc_irs_set		EE VC Set Status Register	Section 8.13.1.45
0x4508	ee_vc_irs_clr		EE VC Clear Status Register	Section 8.13.1.46
0x450C	ee_vc_en_ev0		EE VC Enable EV0 Register	Section 8.13.1.47
0x4510	ee_vc_en_set_ev0		EE VC Enable Set EV0 Register	Section 8.13.1.48
0x4514	ee_vc_en_clr_ev0		EE VC Enable Clear EV0 Register	Section 8.13.1.49
0x4518	ee_vc_en_ev1		EE VC Enable EV1 Register	Section 8.13.1.50
0x451C	ee_vc_en_set_ev1		EE VC Enable Set EV1 Register	Section 8.13.1.51
0x4520	ee_vc_en_clr_ev1		EE VC Enable Clear EV1 Register	Section 8.13.1.52
0x4524	ee_vc_en_sts_ev0		EE VC Enabled Status EV0 Register	Section 8.13.1.53
0x4528	ee_vc_en_sts_ev1		EE VC Enabled Status EV1 Register	Section 8.13.1.54
0x4600	EE_AIF2_RUN_STS		This register is used to indicate the state of AIF2	Section 8.13.1.55
0x4604	EE_AIF2_RUN_CTL		This register is used to indicate the state of AIF2	Section 8.13.1.56
0x4700	ee_err_alrm_orgn		EE Register used to determine which EN_STSregister contains the error/alarm that caused an event	Section 8.13.1.57
0x40000	ee_lk_irs_a[0]	R	EE Link Raw Status A Register	Section 8.13.1.58
0x40004	ee_lk_irs_set_a[0]	W	EE Link Set Status A Register	Section 8.13.1.59
0x40008	ee_lk_irs_clr_a[0]	W	EE Link Clear Status A Register	Section 8.13.1.60
0x4000C	ee_lk_en_a_ev0[0]	R	EE Link Enable A EV0 Register	Section 8.13.1.61
0x40010	ee_lk_en_a_set_ev0[0]	W	EE Link Enable A EV0 Set Register	Section 8.13.1.62
0x40014	ee_lk_en_a_clr_ev0[0]	W	EE Link Enable A EV0 Clear Register	Section 8.13.1.63
0x40018	ee_lk_en_a_ev1[0]	R	EE Link Enable A EV1 Register	Section 8.13.1.64
0x4001C	ee_lk_en_a_set_ev1[0]	W	EE Link Enable A EV1 Set Register	Section 8.13.1.65
0x40020	ee_lk_en_a_clr_ev1[0]	W	EE Link Enable A EV1 Clear Register	Section 8.13.1.66
0x40024	ee_lk_en_sts_a_ev0[0]	R	EE Link Enabled Status A EV0 Register	Section 8.13.1.67
0x40028	ee_lk_en_sts_a_ev1[0]	R	EE Link Enabled Status A EV1 Register	Section 8.13.1.68
0x4002C	ee_lk_irs_b[0]	R	EE Link Raw Status B Register	Section 8.13.1.69
0x40030	ee_lk_irs_set_b[0]	W	EE Link Set Status B Register	Section 8.13.1.70
0x40034	ee_lk_irs_clr_b[0]	W	EE Link Clear Status B Register	Section 8.13.1.71
0x40038	ee_lk_en_b_ev0[0]	R	EE Link Enable B EV0 Register	Section 8.13.1.72
0x4003C	ee_lk_en_b_set_ev0[0]	W	EE Link Enable B EV0 Set Register	Section 8.13.1.73
0x40040	ee_lk_en_b_clr_ev0[0]	W	EE Link Enable B EV0 Clear Register	Section 8.13.1.74
0x40044	ee_lk_en_b_ev1[0]	R	EE Link Enable B EV1 Register	Section 8.13.1.75
0x40048	ee_lk_en_b_set_ev1[0]	W	EE Link Enable B EV1 Set Register	Section 8.13.1.76
0x4004C	ee_lk_en_b_clr_ev1[0]	W	EE Link Enable B EV1 Clear Register	Section 8.13.1.77

Table 8-272. EE Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x40050	ee_lk_en_sts_b_ev0[0]	R	EE Link Enabled Status B EV0 Register	Section 8.13.1.78
0x40054	ee_lk_en_sts_b_ev1[0]	R	EE Link Enabled Status B EV1 Register	Section 8.13.1.79
0x40080	ee_lk_irs_a[1]	R	EE Link Raw Status A Register	Section 8.13.1.58
0x40084	ee_lk_irs_set_a[1]	W	EE Link Set Status A Register	Section 8.13.1.59
0x40088	ee_lk_irs_clr_a[1]	W	EE Link Clear Status A Register	Section 8.13.1.60
0x4008C	ee_lk_en_a_ev0[1]	R	EE Link Enable A EV0 Register	Section 8.13.1.61
0x40090	ee_lk_en_a_set_ev0[1]	W	EE Link Enable A EV0 Set Register	Section 8.13.1.62
0x40094	ee_lk_en_a_clr_ev0[1]	W	EE Link Enable A EV0 Clear Register	Section 8.13.1.63
0x40098	ee_lk_en_a_ev1[1]	R	EE Link Enable A EV1 Register	Section 8.13.1.64
0x4009C	ee_lk_en_a_set_ev1[1]	W	EE Link Enable A EV1 Set Register	Section 8.13.1.65
0x400A0	ee_lk_en_a_clr_ev1[1]	W	EE Link Enable A EV1 Clear Register	Section 8.13.1.66
0x400A4	ee_lk_en_sts_a_ev0[1]	R	EE Link Enabled Status A EV0 Register	Section 8.13.1.67
0x400A8	ee_lk_en_sts_a_ev1[1]	R	EE Link Enabled Status A EV1 Register	Section 8.13.1.68
0x400AC	ee_lk_irs_b[1]	R	EE Link Raw Status B Register	Section 8.13.1.69
0x400B0	ee_lk_irs_set_b[1]	W	EE Link Set Status B Register	Section 8.13.1.70
0x400B4	ee_lk_irs_clr_b[1]	W	EE Link Clear Status B Register	Section 8.13.1.71
0x400B8	ee_lk_en_b_ev0[1]	R	EE Link Enable B EV0 Register	Section 8.13.1.72
0x400BC	ee_lk_en_b_set_ev0[1]	W	EE Link Enable B EV0 Set Register	Section 8.13.1.73
0x400C0	ee_lk_en_b_clr_ev0[1]	W	EE Link Enable B EV0 Clear Register	Section 8.13.1.74
0x400C4	ee_lk_en_b_ev1[1]	R	EE Link Enable B EV1 Register	Section 8.13.1.75
0x400C8	ee_lk_en_b_set_ev1[1]	W	EE Link Enable B EV1 Set Register	Section 8.13.1.76
0x400CC	ee_lk_en_b_clr_ev1[1]	W	EE Link Enable B EV1 Clear Register	Section 8.13.1.77
0x400D0	ee_lk_en_sts_b_ev0[1]	R	EE Link Enabled Status B EV0 Register	Section 8.13.1.78
0x400D4	ee_lk_en_sts_b_ev1[1]	R	EE Link Enabled Status B EV1 Register	Section 8.13.1.79
0x40100	ee_lk_irs_a[2]	R	EE Link Raw Status A Register	Section 8.13.1.58
0x40104	ee_lk_irs_set_a[2]	W	EE Link Set Status A Register	Section 8.13.1.59
0x40108	ee_lk_irs_clr_a[2]	W	EE Link Clear Status A Register	Section 8.13.1.60
0x4010C	ee_lk_en_a_ev0[2]	R	EE Link Enable A EV0 Register	Section 8.13.1.61
0x40110	ee_lk_en_a_set_ev0[2]	W	EE Link Enable A EV0 Set Register	Section 8.13.1.62
0x40114	ee_lk_en_a_clr_ev0[2]	W	EE Link Enable A EV0 Clear Register	Section 8.13.1.63
0x40118	ee_lk_en_a_ev1[2]	R	EE Link Enable A EV1 Register	Section 8.13.1.64
0x4011C	ee_lk_en_a_set_ev1[2]	W	EE Link Enable A EV1 Set Register	Section 8.13.1.65
0x40120	ee_lk_en_a_clr_ev1[2]	W	EE Link Enable A EV1 Clear Register	Section 8.13.1.66
0x40124	ee_lk_en_sts_a_ev0[2]	R	EE Link Enabled Status A EV0 Register	Section 8.13.1.67
0x40128	ee_lk_en_sts_a_ev1[2]	R	EE Link Enabled Status A EV1 Register	Section 8.13.1.68
0x4012C	ee_lk_irs_b[2]	R	EE Link Raw Status B Register	Section 8.13.1.69
0x40130	ee_lk_irs_set_b[2]	W	EE Link Set Status B Register	Section 8.13.1.70
0x40134	ee_lk_irs_clr_b[2]	W	EE Link Clear Status B Register	Section 8.13.1.71
0x40138	ee_lk_en_b_ev0[2]	R	EE Link Enable B EV0 Register	Section 8.13.1.72
0x4013C	ee_lk_en_b_set_ev0[2]	W	EE Link Enable B EV0 Set Register	Section 8.13.1.73
0x40140	ee_lk_en_b_clr_ev0[2]	W	EE Link Enable B EV0 Clear Register	Section 8.13.1.74
0x40144	ee_lk_en_b_ev1[2]	R	EE Link Enable B EV1 Register	Section 8.13.1.75
0x40148	ee_lk_en_b_set_ev1[2]	W	EE Link Enable B EV1 Set Register	Section 8.13.1.76
0x4014C	ee_lk_en_b_clr_ev1[2]	W	EE Link Enable B EV1 Clear Register	Section 8.13.1.77
0x40150	ee_lk_en_sts_b_ev0[2]	R	EE Link Enabled Status B EV0 Register	Section 8.13.1.78
0x40154	ee_lk_en_sts_b_ev1[2]	R	EE Link Enabled Status B EV1 Register	Section 8.13.1.79
0x40180	ee_lk_irs_a[3]	R	EE Link Raw Status A Register	Section 8.13.1.58

Table 8-272. EE Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x40184	ee_lk_irs_set_a[3]	W	EE Link Set Status A Register	Section 8.13.1.59
0x40188	ee_lk_irs_clr_a[3]	W	EE Link Clear Status A Register	Section 8.13.1.60
0x4018C	ee_lk_en_a_ev0[3]	R	EE Link Enable A EV0 Register	Section 8.13.1.61
0x40190	ee_lk_en_a_set_ev0[3]	W	EE Link Enable A EV0 Set Register	Section 8.13.1.62
0x40194	ee_lk_en_a_clr_ev0[3]	W	EE Link Enable A EV0 Clear Register	Section 8.13.1.63
0x40198	ee_lk_en_a_ev1[3]	R	EE Link Enable A EV1 Register	Section 8.13.1.64
0x4019C	ee_lk_en_a_set_ev1[3]	W	EE Link Enable A EV1 Set Register	Section 8.13.1.65
0x401A0	ee_lk_en_a_clr_ev1[3]	W	EE Link Enable A EV1 Clear Register	Section 8.13.1.66
0x401A4	ee_lk_en_sts_a_ev0[3]	R	EE Link Enabled Status A EV0 Register	Section 8.13.1.67
0x401A8	ee_lk_en_sts_a_ev1[3]	R	EE Link Enabled Status A EV1 Register	Section 8.13.1.68
0x401AC	ee_lk_irs_b[3]	R	EE Link Raw Status B Register	Section 8.13.1.69
0x401B0	ee_lk_irs_set_b[3]	W	EE Link Set Status B Register	Section 8.13.1.70
0x401B4	ee_lk_irs_clr_b[3]	W	EE Link Clear Status B Register	Section 8.13.1.71
0x401B8	ee_lk_en_b_ev0[3]	R	EE Link Enable B EV0 Register	Section 8.13.1.72
0x401BC	ee_lk_en_b_set_ev0[3]	W	EE Link Enable B EV0 Set Register	Section 8.13.1.73
0x401C0	ee_lk_en_b_clr_ev0[3]	W	EE Link Enable B EV0 Clear Register	Section 8.13.1.74
0x401C4	ee_lk_en_b_ev1[3]	R	EE Link Enable B EV1 Register	Section 8.13.1.75
0x401C8	ee_lk_en_b_set_ev1[3]	W	EE Link Enable B EV1 Set Register	Section 8.13.1.76
0x401CC	ee_lk_en_b_clr_ev1[3]	W	EE Link Enable B EV1 Clear Register	Section 8.13.1.77
0x401D0	ee_lk_en_sts_b_ev0[3]	R	EE Link Enabled Status B EV0 Register	Section 8.13.1.78
0x401D4	ee_lk_en_sts_b_ev1[3]	R	EE Link Enabled Status B EV1 Register	Section 8.13.1.79
0x40200	ee_lk_irs_a[4]	R	EE Link Raw Status A Register	Section 8.13.1.58
0x40204	ee_lk_irs_set_a[4]	W	EE Link Set Status A Register	Section 8.13.1.59
0x40208	ee_lk_irs_clr_a[4]	W	EE Link Clear Status A Register	Section 8.13.1.60
0x4020C	ee_lk_en_a_ev0[4]	R	EE Link Enable A EV0 Register	Section 8.13.1.61
0x40210	ee_lk_en_a_set_ev0[4]	W	EE Link Enable A EV0 Set Register	Section 8.13.1.62
0x40214	ee_lk_en_a_clr_ev0[4]	W	EE Link Enable A EV0 Clear Register	Section 8.13.1.63
0x40218	ee_lk_en_a_ev1[4]	R	EE Link Enable A EV1 Register	Section 8.13.1.64
0x4021C	ee_lk_en_a_set_ev1[4]	W	EE Link Enable A EV1 Set Register	Section 8.13.1.65
0x40220	ee_lk_en_a_clr_ev1[4]	W	EE Link Enable A EV1 Clear Register	Section 8.13.1.66
0x40224	ee_lk_en_sts_a_ev0[4]	R	EE Link Enabled Status A EV0 Register	Section 8.13.1.67
0x40228	ee_lk_en_sts_a_ev1[4]	R	EE Link Enabled Status A EV1 Register	Section 8.13.1.68
0x4022C	ee_lk_irs_b[4]	R	EE Link Raw Status B Register	Section 8.13.1.69
0x40230	ee_lk_irs_set_b[4]	W	EE Link Set Status B Register	Section 8.13.1.70
0x40234	ee_lk_irs_clr_b[4]	W	EE Link Clear Status B Register	Section 8.13.1.71
0x40238	ee_lk_en_b_ev0[4]	R	EE Link Enable B EV0 Register	Section 8.13.1.72
0x4023C	ee_lk_en_b_set_ev0[4]	W	EE Link Enable B EV0 Set Register	Section 8.13.1.73
0x40240	ee_lk_en_b_clr_ev0[4]	W	EE Link Enable B EV0 Clear Register	Section 8.13.1.74
0x40244	ee_lk_en_b_ev1[4]	R	EE Link Enable B EV1 Register	Section 8.13.1.75
0x40248	ee_lk_en_b_set_ev1[4]	W	EE Link Enable B EV1 Set Register	Section 8.13.1.76
0x4024C	ee_lk_en_b_clr_ev1[4]	W	EE Link Enable B EV1 Clear Register	Section 8.13.1.77
0x40250	ee_lk_en_sts_b_ev0[4]	R	EE Link Enabled Status B EV0 Register	Section 8.13.1.78
0x40254	ee_lk_en_sts_b_ev1[4]	R	EE Link Enabled Status B EV1 Register	Section 8.13.1.79
0x40280	ee_lk_irs_a[5]	R	EE Link Raw Status A Register	Section 8.13.1.58
0x40284	ee_lk_irs_set_a[5]	W	EE Link Set Status A Register	Section 8.13.1.59
0x40288	ee_lk_irs_clr_a[5]	W	EE Link Clear Status A Register	Section 8.13.1.60
0x4028C	ee_lk_en_a_ev0[5]	R	EE Link Enable A EV0 Register	Section 8.13.1.61

Table 8-272. EE Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x40290	ee_lk_en_a_set_ev0[5]	W	EE Link Enable A EV0 Set Register	Section 8.13.1.62
0x40294	ee_lk_en_a_clr_ev0[5]	W	EE Link Enable A EV0 Clear Register	Section 8.13.1.63
0x40298	ee_lk_en_a_ev1[5]	R	EE Link Enable A EV1 Register	Section 8.13.1.64
0x4029C	ee_lk_en_a_set_ev1[5]	W	EE Link Enable A EV1 Set Register	Section 8.13.1.65
0x402A0	ee_lk_en_a_clr_ev1[5]	W	EE Link Enable A EV1 Clear Register	Section 8.13.1.66
0x402A4	ee_lk_en_sts_a_ev0[5]	R	EE Link Enabled Status A EV0 Register	Section 8.13.1.67
0x402A8	ee_lk_en_sts_a_ev1[5]	R	EE Link Enabled Status A EV1 Register	Section 8.13.1.68
0x402AC	ee_lk_irs_b[5]	R	EE Link Raw Status B Register	Section 8.13.1.69
0x402B0	ee_lk_irs_set_b[5]	W	EE Link Set Status B Register	Section 8.13.1.70
0x402B4	ee_lk_irs_clr_b[5]	W	EE Link Clear Status B Register	Section 8.13.1.71
0x402B8	ee_lk_en_b_ev0[5]	R	EE Link Enable B EV0 Register	Section 8.13.1.72
0x402BC	ee_lk_en_b_set_ev0[5]	W	EE Link Enable B EV0 Set Register	Section 8.13.1.73
0x402C0	ee_lk_en_b_clr_ev0[5]	W	EE Link Enable B EV0 Clear Register	Section 8.13.1.74
0x402C4	ee_lk_en_b_ev1[5]	R	EE Link Enable B EV1 Register	Section 8.13.1.75
0x402C8	ee_lk_en_b_set_ev1[5]	W	EE Link Enable B EV1 Set Register	Section 8.13.1.76
0x402CC	ee_lk_en_b_clr_ev1[5]	W	EE Link Enable B EV1 Clear Register	Section 8.13.1.77
0x402D0	ee_lk_en_sts_b_ev0[5]	R	EE Link Enabled Status B EV0 Register	Section 8.13.1.78
0x402D4	ee_lk_en_sts_b_ev1[5]	R	EE Link Enabled Status B EV1 Register	Section 8.13.1.79
0x40300	ee_at_irs		EE AT Raw Status Register	Section 8.13.1.80
0x40304	ee_at_irs_set		EE AT Set Status Register	Section 8.13.1.81
0x40308	ee_at_irs_clr		EE AT Clear Status Register	Section 8.13.1.82
0x4030C	ee_at_en_ev0		EE AT Enable EVO Register	Section 8.13.1.83
0x40310	ee_at_en_set_ev0		EE AT Enable Set EV0 Register	Section 8.13.1.84
0x40314	ee_at_en_clr_ev0		EE AT Enable Clear EV0 Register	Section 8.13.1.85
0x40318	ee_at_en_ev1		EE AT Enable EV1 Register	Section 8.13.1.86
0x4031C	ee_at_en_set_ev1		EE AT Enable Set EV1 Register	Section 8.13.1.87
0x40320	ee_at_en_clr_ev1		EE AT Enable Clear EV1 Register	Section 8.13.1.88
0x40324	ee_at_en_sts_ev0		EE AT Enabled Status EV0 Register	Section 8.13.1.89
0x40328	ee_at_en_sts_ev1		EE AT Enabled Status EV1 Register	Section 8.13.1.90
0x40400	ee_pd_common_irs		EE PD Common Raw Status Register	Section 8.13.1.91
0x40404	ee_pd_common_irs_set		EE PD Common Set Status Register	Section 8.13.1.92
0x40408	ee_pd_common_irs_clr		EE PD Common Clear Status Register	Section 8.13.1.93
0x4040C	ee_pd_common_en_ev0		EE PD Common Enable EVO Register	Section 8.13.1.94
0x40410	ee_pd_common_en_set_ev0		EE PD Common Enable Set EV0 Register	Section 8.13.1.95
0x40414	ee_pd_common_en_clr_ev0		EE PD Common Enable Clear EV0 Register	Section 8.13.1.96
0x40418	ee_pd_common_en_ev1		EE PD Common Enable EV1 Register	Section 8.13.1.97
0x4041C	ee_pd_common_en_set_ev1		EE PD Common Enable Set EV1 Register	Section 8.13.1.98
0x40420	ee_pd_common_en_clr_ev1		EE PD Common Enable Clear EV1 Register	Section 8.13.1.99
0x40424	ee_pd_common_en_sts_ev0		EE PD Common Enabled Status EV0 Register	Section 8.13.1.100
0x40428	ee_pd_common_en_sts_ev1		EE PD Common Enabled Status EV1 Register	Section 8.13.1.101
0x40400	ee_pd_common_irs		EE PE Common Raw Status Register	Section 8.13.1.102
0x40504	ee_pe_common_irs_set		EE PE Common Set Status Register	Section 8.13.1.103
0x40508	ee_pe_common_irs_clr		EE PE Common Clear Status Register	Section 8.13.1.104
0x4050C	ee_pe_common_en_ev0		EE PE Common Enable EVO Register	Section 8.13.1.105
0x40510	ee_pe_common_en_set_ev0		EE PE Common Enable Set EV0 Register	Section 8.13.1.106
0x40514	ee_pe_common_en_clr_ev0		EE PE Common Enable Clear EV0 Register	Section 8.13.1.107

Table 8-272. EE Register Memory Map (continued)

Offset	Register Name	Access	Description	Section
0x40518	ee_pe_common_en_ev1		EE PE Common Enable EV1 Register	Section 8.13.1.108
0x4051C	ee_pe_common_en_set_ev1		EE PE Common Enable Set EV1 Register	Section 8.13.1.109
0x40520	ee_pe_common_en_clr_ev1		EE PE Common Enable Clear EV1 Register	Section 8.13.1.110
0x40524	ee_pe_common_en_sts_ev0		EE PE Common Enabled Status EV0 Register	Section 8.13.1.111
0x40528	ee_pe_common_en_sts_ev1		EE PE Common Enabled Status EV1 Register	Section 8.13.1.112

8.13.1 Grouped Common Registers Details

8.13.1.1 EE_VB_EOI Register (offset = 0x4000)

Table 8-273. EE_VB_EOI Register Field Descriptions

Bits	Field Name	Type	Description
31-8	RSVD	NOT_ACCESSIBLE	Reserved.
7-0	eoi_vector	READ_WRITE	End of interrupt vector value that distinguishes which interrupt is being acknowledged. This value is outputted on the external eoi_vector interface of the AIF2.

8.13.1.2 EE_VB_INTR_SET Register (offset = 0x4004)

Table 8-274. EE_VB_INTR_SET Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	Reserved.
2	cdma_intr_set	WRITE	AIF2 PKTDMA Interrupt Set. 0 = No Action 1 = Set the external AIF2 PKTDMA Interrupt signal to a 1 (high)
1	alarm_intr_set	WRITE	AIF2 Alarm Interrupt Set. 0 = No Action 1 = Set the external AIF2 Alarm Interrupt signal to a 1 (high)
0	err_intr_set	WRITE	AIF2 Error Interrupt Set. 0 = No Action 1 = Set the external AIF2 Error Interrupt signal to a 1 (high)

8.13.1.3 EE_VB_INTR_CLR Register (offset = 0x4008)

Table 8-275. EE_VB_INTR_CLR Register Field Descriptions

Bits	Field Name	Type	Description
31-3	RSVD	NOT_ACCESSIBLE	Reserved.
2	cdma_intr_clr	WRITE	AIF2 PKTDMA Interrupt Clear. 0 = No Action 1 = Clear the external AIF2 PKTDMA Interrupt signal to a 0 (low)
1	alarm_intr_clr	WRITE	AIF2 Alarm Interrupt Clear. 0 = No Action 1 = Clear the external AIF2 Alarm Interrupt signal to a 0 (low)
0	err_intr_clr	WRITE	AIF2 Error Interrupt Clear. 0 = No Action 1 = Clear the external AIF2 Error Interrupt signal to a 0 (low)

8.13.1.4 ee_db_irs Register (offset = 0x4100)

Table 8-276. ee_db_irs Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	READ	(**Error**) EDB received data from PKTDMA with undefined Multicore Navigator data type. This is an indication of an application error. This event occurs if the Multicore Navigator Data Type received from the PKTDMA is not one of the ones supported.
5	db_ee_e_cd_data_err	READ	(**Information**) Activity monitor that is active each time the Egress DB receives data from the PKTDMA
4	db_ee_e_ps_axc_err	READ	(**Error**) AxC in Multicore Navigator Protocol Specific data does not match Multicore Navigator thread id. This is an indication of an application error. This event occurs if the application put data for an AxC in the wrong buffer in external memory
3	db_ee_i_pd2db_full_err	READ	(**Error**) PD-to-DB Bridge full. This is a catastrophic condition and indicates that the VBUS clock rate is too slow. Although this bridge is 64-deep, the error event will first occur when the bridge depth is 56 as an early warning
2	db_ee_i_fifo_ovfl_err	READ	(**Information**) Ingress FIFO buffer channel overflowed. This can happen if the AIF2 VBUSM interface is stalled by the system. This is not a fatal error. The AIF2 can gracefully recover from this condition
1	db_ee_i_token_ovfl_err	READ	(**Error**) Ingress Packet Data Token FIFO overflowed. This is a catastrophic condition and software needs to effectively flush all of the data buffers in the Ingress DB RAM and re-start the PD
0	db_ee_i_trc_ram_ovfl_err	READ	(**Error**) Data Trace RAM overflowed. This is not a fatal error because it only affects the Data Trace RAM

8.13.1.5 ee_db_irs_set Register (offset = 0x4104)

Table 8-277. ee_db_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	WRITE	DB Engress PKTDMA Data Type Error Set.
5	db_ee_e_cd_data_err	WRITE	DB Egress PKTDMA Data Error Set.
4	db_ee_e_ps_axc_err	WRITE	DB Egress AXC Token Overflow Error Set.
3	db_ee_i_pd2db_full_err	WRITE	DB Ingress PM Token Overflow Error Set.
2	db_ee_i_fifo_ovfl_err	WRITE	DB Ingress FIFO Overflow Error Set.
1	db_ee_i_token_ovfl_err	WRITE	DB Ingress Token Overflow Error Set.
0	db_ee_i_trc_ram_ovfl_err	WRITE	DB Ingress TRC Token Overflow Error Set.

8.13.1.6 ee_db_irs_clr Register (offset = 0x4108)

Table 8-278. ee_db_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	WRITE	DB Egress PKTDMA Data Type Error Clear.
5	db_ee_e_cd_data_err	WRITE	DB Egress PKTDMA Data Error Clear.
4	db_ee_e_ps_axc_err	WRITE	DB Egress AXC Token Overflow Error Clear.
3	db_ee_i_pd2db_full_err	WRITE	DB Ingress PM Token Overflow Error Clear.
2	db_ee_i_fifo_ovfl_err	WRITE	DB Ingress FIFO Overflow Error Clear.
1	db_ee_i_token_ovfl_err	WRITE	DB Ingress Token Overflow Error Clear.
0	db_ee_i_trc_ram_ovfl_err	WRITE	DB Ingress TRC Token Overflow Error Clear.

8.13.1.7 ee_db_en_ev0 Register (offset = 0x410C)

Table 8-279. ee_db_en_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	READ	DB Egress PKTDMA Data Type Error Enable.
5	db_ee_e_cd_data_err	READ	DB Egress PKTDMA Data Error Enable.
4	db_ee_e_ps_axc_err	READ	DB Egress AXC Token Overflow Error Enable.
3	db_ee_i_pd2db_full_err	READ	DB Ingress PM Token Overflow Error Enable.
2	db_ee_i_fifo_ovfl_err	READ	DB Ingress FIFO Overflow Error Enable.
1	db_ee_i_token_ovfl_err	READ	DB Ingress Token Overflow Error Enable.
0	db_ee_i_trc_ram_ovfl_err	READ	DB Ingress TRC Token Overflow Error Enable.

8.13.1.8 ee_db_en_set_ev0 Register (offset = 0x4110)

Table 8-280. ee_db_en_set_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	WRITE	DB Egress PKTDMA Data Type Error Enable Set.
5	db_ee_e_cd_data_err	WRITE	DB Egress PKTDMA Data Error Enable Set.
4	db_ee_e_ps_axc_err	WRITE	DB Egress AXC Token Overflow Error Enable Set.
3	db_ee_i_pd2db_full_err	WRITE	DB Ingress PM Token Overflow Error Enable Set.
2	db_ee_i_fifo_ovfl_err	WRITE	DB Ingress FIFO Overflow Error Enable Set.
1	db_ee_i_token_ovfl_err	WRITE	DB Ingress Token Overflow Error Enable Set.
0	db_ee_i_trc_ram_ovfl_err	WRITE	DB Ingress TRC Token Overflow Error Enable Set.

8.13.1.9 ee_db_en_clr_ev0 Register (offset = 0x4114)

Table 8-281. ee_db_en_clr_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	WRITE	DB Egress PKTDMA Data Type Error Enable Clear.
5	db_ee_e_cd_data_err	WRITE	DB Egress PKTDMA Data Error Enable Clear.
4	db_ee_e_ps_axc_err	WRITE	DB Egress AXC Token Overflow Error Enable Clear.
3	db_ee_i_pd2db_full_err	WRITE	DB Ingress PM Token Overflow Error Enable Clear.
2	db_ee_i_fifo_ovfl_err	WRITE	DB Ingress FIFO Overflow Error Enable Clear.
1	db_ee_i_token_ovfl_err	WRITE	DB Ingress Token Overflow Error Enable Clear.
0	db_ee_i_trc_ram_ovfl_err	WRITE	DB Ingress TRC Token Overflow Error Enable Clear.

8.13.1.10 ee_db_en_ev1 Register (offset = 0x4118)

Table 8-282. ee_db_en_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	READ	DB Egress PKTDMA Data Type Error Enable.
5	db_ee_e_cd_data_err	READ	DB Egress PKTDMA Data Error Enable.
4	db_ee_e_ps_axc_err	READ	DB Egress AXC Token Overflow Error Enable.
3	db_ee_i_pd2db_full_err	READ	DB Ingress PM Token Overflow Error Enable.
2	db_ee_i_fifo_ovfl_err	READ	DB Ingress FIFO Overflow Error Enable.
1	db_ee_i_token_ovfl_err	READ	DB Ingress Token Overflow Error Enable.
0	db_ee_i_trc_ram_ovfl_err	READ	DB Ingress TRC Token Overflow Error Enable.

8.13.1.11 ee_db_en_set_ev1 Register (offset = 0x411C)

Table 8-283. ee_db_en_set_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	WRITE	DB Egress PKTDMA Data Type Error Enable Set.
5	db_ee_e_cd_data_err	WRITE	DB Egress PKTDMA Data Error Enable Set.
4	db_ee_e_ps_axc_err	WRITE	DB Egress AXC Token Overflow Error Enable Set.
3	db_ee_i_pd2db_full_err	WRITE	DB Ingress PM Token Overflow Error Enable Set.
2	db_ee_i_fifo_ovfl_err	WRITE	DB Ingress FIFO Overflow Error Enable Set.
1	db_ee_i_token_ovfl_err	WRITE	DB Ingress Token Overflow Error Enable Set.
0	db_ee_i_trc_ram_ovfl_err	WRITE	DB Ingress TRC Token Overflow Error Enable Set.

8.13.1.12 ee_db_en_clr_ev1 Register (offset = 0x4120)

Table 8-284. ee_db_en_clr_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	WRITE	DB Egress PKTDMA Data Type Error Enable Clear.
5	db_ee_e_cd_data_err	WRITE	DB Egress PKTDMA Data Error Enable Clear.
4	db_ee_e_ps_axc_err	WRITE	DB Egress AXC Token Overflow Error Enable Clear.
3	db_ee_i_pd2db_full_err	WRITE	DB Ingress PM Token Overflow Error Enable Clear.
2	db_ee_i_fifo_ovfl_err	WRITE	DB Ingress FIFO Overflow Error Enable Clear.
1	db_ee_i_token_ovfl_err	WRITE	DB Ingress Token Overflow Error Enable Clear.
0	db_ee_i_trc_ram_ovfl_err	WRITE	DB Ingress TRC Token Overflow Error Enable Clear.

8.13.1.13 ee_db_en_sts_ev0 Register (offset = 0x4124)

Table 8-285. ee_db_en_sts_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	READ	DB Egress PKTDMA Data Type Error Enabled.
5	db_ee_e_cd_data_err	READ	DB Egress PKTDMA Data Error Enabled.
4	db_ee_e_ps_axc_err	READ	DB Egress AXC Token Overflow Error Enabled.
3	db_ee_i_pd2db_full_err	READ	DB Ingress PM Token Overflow Error Enabled.
2	db_ee_i_fifo_ovfl_err	READ	DB Ingress FIFO Overflow Error Enabled.
1	db_ee_i_token_ovfl_err	READ	DB Ingress Token Overflow Error Enabled.
0	db_ee_i_trc_ram_ovfl_err	READ	DB Ingress TRC Token Overflow Error Enabled.

8.13.1.14 ee_db_en_sts_ev1 Register (offset = 0x4128)

Table 8-286. ee_db_en_sts_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-7	RSVD	NOT_ACCESSIBLE	Reserved.
6	db_ee_e_cd_data_type_err	READ	DB Egress PKTDMA Data Type Error Enabled.
5	db_ee_e_cd_data_err	READ	DB Egress PKTDMA Data Error Enabled.
4	db_ee_e_ps_axc_err	READ	DB Egress AXC Token Overflow Error Enabled.
3	db_ee_i_pd2db_full_err	READ	DB Ingress PM Token Overflow Error Enabled.
2	db_ee_i_fifo_ovfl_err	READ	DB Ingress FIFO Overflow Error Enabled.
1	db_ee_i_token_ovfl_err	READ	DB Ingress Token Overflow Error Enabled.
0	db_ee_i_trc_ram_ovfl_err	READ	DB Ingress TRC Token Overflow Error Enabled.

8.13.1.15 ee_ad_irs Register (offset = 0x4200)

Table 8-287. ee_ad_irs Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	READ	(**Error**) For each of three Egress DMA channels, if a trigger request is received while a request for that channel is still pending, a DMA error event will occur.
25	ad_ee_e_dma_1_err	READ	(**Error**) For each of three Egress DMA channels, if a trigger request is received while a request for that channel is still pending, a DMA error event will occur
24	ad_ee_e_dma_0_err	READ	(**Error**) For each of three Egress DMA channels, if a trigger request is received while a request for that channel is still pending, a DMA error event will occur
18	ad_ee_i_dma_2_err	READ	(**Error**) For each of three Ingress DMA channels, if a trigger request is received while a request for that channel is still pending, a DMA error event will occur.
17	ad_ee_i_dma_1_err	READ	(**Error**) For each of three Ingress DMA channels, if a trigger request is received while a request for that channel is still pending, a DMA error event will occur
16	ad_ee_i_dma_0_err	READ	(**Error**) For each of three Ingress DMA channels, if a trigger request is received while a request for that channel is still pending, a DMA error event will occur
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	READ	(**Information**) Activity monitor that is active high each time the Egress AD Scheduler sends a request to read data to the PKTDMA.
0	ad_ee_i_cd_data_err	READ	(**Information**) Activity monitor that is active whenever the Ingress AD sends data to the PKTDMA.

8.13.1.16 ee_ad_irs_set Register (offset = 0x4204)

Table 8-288. ee_ad_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	WRITE	AD Egress DMA Bit 2 Error Set.
25	ad_ee_e_dma_1_err	WRITE	AD Egress DMA Bit 1 Error Set.
24	ad_ee_e_dma_0_err	WRITE	AD Egress DMA Bit 0 Error Set.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	WRITE	AD Ingress DMA Bit 2 Error Set.
17	ad_ee_i_dma_1_err	WRITE	AD Ingress DMA Bit 1 Error Set.
16	ad_ee_i_dma_0_err	WRITE	AD Ingress DMA Bit 0 Error Set.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	WRITE	AD Engress PKTDMA Scheduler Error Set.
0	ad_ee_i_cd_data_err	WRITE	AD Egress PKTDMA Data Error Set.

8.13.1.17 ee_ad_irs_clr Register (offset = 0x4208)

Table 8-289. ee_ad_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	WRITE	AD Egress DMA Bit 2 Error Clear.
25	ad_ee_e_dma_1_err	WRITE	AD Egress DMA Bit 1 Error Clear.
24	ad_ee_e_dma_0_err	WRITE	AD Egress DMA Bit 0 Error Clear.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	WRITE	AD Ingress DMA Bit 2 Error Clear.
17	ad_ee_i_dma_1_err	WRITE	AD Ingress DMA Bit 1 Error Clear.
16	ad_ee_i_dma_0_err	WRITE	AD Ingress DMA Bit 0 Error Clear.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	WRITE	AD Engress PKTDMA Scheduler Error Clear.
0	ad_ee_i_cd_data_err	WRITE	AD Egress PKTDMA Data Error Clear.

8.13.1.18 ee_ad_en_ev0 Register (offset = 0x420C)

Table 8-290. ee_ad_en_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	READ	AD Egress DMA Bit 2 Error Enable.
25	ad_ee_e_dma_1_err	READ	AD Egress DMA Bit 1 Error Enable.
24	ad_ee_e_dma_0_err	READ	AD Egress DMA Bit 0 Error Enable.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	READ	AD Ingress DMA Bit 2 Error Enable.
17	ad_ee_i_dma_1_err	READ	AD Ingress DMA Bit 1 Error Enable.
16	ad_ee_i_dma_0_err	READ	AD Ingress DMA Bit 0 Error Enable.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	READ	AD Engress PKTDMA Scheduler Error Enable.
0	ad_ee_i_cd_data_err	READ	AD Egress PKTDMA Data Error Enable.

8.13.1.19 ee_ad_en_set_ev0 Register (0x4210)

Table 8-291. ee_ad_en_set_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	WRITE	AD Egress DMA Bit 2 Error Enable Set.
25	ad_ee_e_dma_1_err	WRITE	AD Egress DMA Bit 1 Error Enable Set.
24	ad_ee_e_dma_0_err	WRITE	AD Egress DMA Bit 0 Error Enable Set.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	WRITE	AD Ingress DMA Bit 2 Error Enable Set.
17	ad_ee_i_dma_1_err	WRITE	AD Ingress DMA Bit 1 Error Enable Set.
16	ad_ee_i_dma_0_err	WRITE	AD Ingress DMA Bit 0 Error Enable Set.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	WRITE	AD Engress PKTDMA Scheduler Error Enable Set.
0	ad_ee_i_cd_data_err	WRITE	AD Egress PKTDMA Data Error Enable Set.

8.13.1.20 ee_ad_en_clr_ev0 Register (offset = 0x4214)

Table 8-292. ee_ad_en_clr_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	WRITE	AD Egress DMA Bit 2 Error Enable Clear.
25	ad_ee_e_dma_1_err	WRITE	AD Egress DMA Bit 1 Error Enable Clear.
24	ad_ee_e_dma_0_err	WRITE	AD Egress DMA Bit 0 Error Enable Clear.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	WRITE	AD Ingress DMA Bit 2 Error Enable Clear.
17	ad_ee_i_dma_1_err	WRITE	AD Ingress DMA Bit 1 Error Enable Clear.
16	ad_ee_i_dma_0_err	WRITE	AD Ingress DMA Bit 0 Error Enable Clear.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	WRITE	AD Engress PKTDMA Scheduler Error Enable Clear.
0	ad_ee_i_cd_data_err	WRITE	AD Egress PKTDMA Data Error Enable Clear.

8.13.1.21 ee_ad_en_ev1 Register (offset = 0x4218)

Table 8-293. ee_ad_en_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	READ	AD Egress DMA Bit 2 Error Enable.
25	ad_ee_e_dma_1_err	READ	AD Egress DMA Bit 1 Error Enable.
24	ad_ee_e_dma_0_err	READ	AD Egress DMA Bit 0 Error Enable.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	READ	AD Ingress DMA Bit 2 Error Enable.
17	ad_ee_i_dma_1_err	READ	AD Ingress DMA Bit 1 Error Enable.
16	ad_ee_i_dma_0_err	READ	AD Ingress DMA Bit 0 Error Enable.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	READ	AD Engress PKTDMA Scheduler Error Enable.
0	ad_ee_i_cd_data_err	READ	AD Egress PKTDMA Data Error Enable.

8.13.1.22 ee_ad_en_set_ev1 Register (offset = 0x421C)

Table 8-294. ee_ad_en_set_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	WRITE	AD Egress DMA Bit 2 Error Enable Set.
25	ad_ee_e_dma_1_err	WRITE	AD Egress DMA Bit 1 Error Enable Set.
24	ad_ee_e_dma_0_err	WRITE	AD Egress DMA Bit 0 Error Enable Set.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	WRITE	AD Ingress DMA Bit 2 Error Enable Set.
17	ad_ee_i_dma_1_err	WRITE	AD Ingress DMA Bit 1 Error Enable Set.
16	ad_ee_i_dma_0_err	WRITE	AD Ingress DMA Bit 0 Error Enable Set.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	WRITE	AD Engress PKTDMA Scheduler Error Enable Set.
0	ad_ee_i_cd_data_err	WRITE	AD Egress PKTDMA Data Error Enable Set.

8.13.1.23 ee_ad_en_clr_ev1 Register (offset = 0x4220)

Table 8-295. ee_ad_en_clr_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	WRITE	AD Egress DMA Bit 2 Error Enable Clear.
25	ad_ee_e_dma_1_err	WRITE	AD Egress DMA Bit 1 Error Enable Clear.
24	ad_ee_e_dma_0_err	WRITE	AD Egress DMA Bit 0 Error Enable Clear.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	WRITE	AD Ingress DMA Bit 2 Error Enable Clear.
17	ad_ee_i_dma_1_err	WRITE	AD Ingress DMA Bit 1 Error Enable Clear.
16	ad_ee_i_dma_0_err	WRITE	AD Ingress DMA Bit 0 Error Enable Clear.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	WRITE	AD Engress PKTDMA Scheduler Error Enable Clear.
0	ad_ee_i_cd_data_err	WRITE	AD Egress PKTDMA Data Error Enable Clear.

8.13.1.24 ee_ad_en_sts_ev0 Register (offset = 0x4224)

Table 8-296. ee_ad_en_sts_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	READ	AD Egress DMA Bit 2 Error Enabled.
25	ad_ee_e_dma_1_err	READ	AD Egress DMA Bit 1 Error Enabled.
24	ad_ee_e_dma_0_err	READ	AD Egress DMA Bit 0 Error Enabled.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	READ	AD Ingress DMA Bit 2 Error Enabled.
17	ad_ee_i_dma_1_err	READ	AD Ingress DMA Bit 1 Error Enabled.
16	ad_ee_i_dma_0_err	READ	AD Ingress DMA Bit 0 Error Enabled.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	READ	AD Engress PKTDMA Scheduler Error Enabled.
0	ad_ee_i_cd_data_err	READ	AD Egress PKTDMA Data Error Enabled.

8.13.1.25 ee_ad_en_sts_ev1 Register (offset = 0x4228)

Table 8-297. ee_ad_en_sts_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-27	RSVD2	NOT_ACCESSIBLE	Reserved.
26	ad_ee_e_dma_2_err	READ	AD Egress DMA Bit 2 Error Enabled.
25	ad_ee_e_dma_1_err	READ	AD Egress DMA Bit 1 Error Enabled.
24	ad_ee_e_dma_0_err	READ	AD Egress DMA Bit 0 Error Enabled.
23-19	RSVD1	NOT_ACCESSIBLE	Reserved.
18	ad_ee_i_dma_2_err	READ	AD Ingress DMA Bit 2 Error Enabled.
17	ad_ee_i_dma_1_err	READ	AD Ingress DMA Bit 1 Error Enabled.
16	ad_ee_i_dma_0_err	READ	AD Ingress DMA Bit 0 Error Enabled.
15-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	ad_ee_e_cd_sch_err	READ	AD Engress PKTDMA Scheduler Error Enabled.
0	ad_ee_i_cd_data_err	READ	AD Egress PKTDMA Data Error Enabled.

8.13.1.26 ee_cd_irs Register (offset = 0x4300)

Table 8-298. ee_cd_irs Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	cd_ee_mop_desc_starve_err	READ	(**Error**) PKTDMA MOP Descriptor Starvation
0	cd_ee_sop_desc_starve_err	READ	(**Error**) PKTDMA SOP Descriptor Starvation

8.13.1.27 ee_cd_irs_set Register (offset = 0x4304)

Table 8-299. ee_cd_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	cd_ee_mop_desc_starve_err	WRITE	CD MOP Desc. Starve Error Set.
0	cd_ee_sop_desc_starve_err	WRITE	CD SOP Desc. Starve Error Set.

8.13.1.28 ee_cd_irs_clr Register (offset = 0x4308)

Table 8-300. ee_cd_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	cd_ee_mop_desc_starve_err	WRITE	CD MOP Desc. Starve Error Clear.
0	cd_ee_sop_desc_starve_err	WRITE	CD SOP Desc. Starve Error Clear.

8.13.1.29 ee_cd_en_ev Register (offset = 0x430C)

Table 8-301. ee_cd_en_ev Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	cd_ee_mop_desc_starve_err	READ	CD MOP Desc. Starve Error Enable.
0	cd_ee_sop_desc_starve_err	READ	CD SOP Desc. Starve Error Enable.

8.13.1.30 ee_cd_en_set_ev Register (offset = 0x4310)

Table 8-302. ee_cd_en_set_ev Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	cd_ee_mop_desc_starve_err	WRITE	CD MOP Desc. Starve Error Enable Set.
0	cd_ee_sop_desc_starve_err	WRITE	CD SOP Desc. Starve Error Enable Set.

8.13.1.31 ee_cd_en_clr_ev Register (offset = x4314)

Table 8-303. ee_cd_en_clr_ev Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	cd_ee_mop_desc_starve_err	WRITE	CD MOP Desc. Starve Error Enable Clear.
0	cd_ee_sop_desc_starve_err	WRITE	CD SOP Desc. Starve Error Enable Clear.

8.13.1.32 ee_cd_en_sts_ev Register (offset = 0x4318)

Table 8-304. ee_cd_en_sts_ev Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	cd_ee_mop_desc_starve_err	READ	CD MOP Desc. Starve Error Enabled.
0	cd_ee_sop_desc_starve_err	READ	CD SOP Desc. Starve Error Enabled.

8.13.1.33 ee_sd_irs Register (offset = 0x4400)

Table 8-305. ee_sd_irs Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	READ	(**Information**) If B8 PLL is locked correctly, this field will be set
0	sd_ee_stspll_b4_err	READ	(**Information**) If B4 PLL is locked correctly, this field will be set

8.13.1.34 ee_sd_irs_set Register (offset = 0x4404)

Table 8-306. ee_sd_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	WRITE	SD B8 PLL Error Set.
0	sd_ee_stspll_b4_err	WRITE	SD B4 PLL Error Set.

8.13.1.35 ee_sd_irs_clr Register (offset = 0x4408)

Table 8-307. ee_sd_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	WRITE	SD B8 PLL Error Clear
0	sd_ee_stspll_b4_err	WRITE	SD B4 PLL Error Clear

8.13.1.36 ee_sd_en_ev0 Register (offset = 0x440C)

Table 8-308. ee_sd_en_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	READ	SD B8 PLL Error Enable.
0	sd_ee_stspll_b4_err	READ	SD B4 PLL Error Enable.

8.13.1.37 ee_sd_en_set_ev0 Register (offset = 0x4410)

Table 8-309. ee_sd_en_set_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	WRITE	SD B8 PLL Error Set Enable.
0	sd_ee_stspll_b4_err	WRITE	SD B4 PLL Error Set Enable.

8.13.1.38 ee_sd_en_clr_ev0 Register (offset = 0x4414)

Table 8-310. ee_sd_en_clr_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	WRITE	SD B8 PLL Error Clear Enable.
0	sd_ee_stspll_b4_err	WRITE	SD B4 PLL Error Clear Enable.

8.13.1.39 ee_sd_en_ev1 Register (offset = 0x4418)

Table 8-311. ee_sd_en_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	READ	SD B8 PLL Error Enable.
0	sd_ee_stspll_b4_err	READ	SD B4 PLL Error Enable.

8.13.1.40 ee_sd_en_set_ev1 Register (offset = 0x441C)

Table 8-312. ee_sd_en_set_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	WRITE	SD B8 PLL Error Set Enable.
0	sd_ee_stspll_b4_err	WRITE	SD B4 PLL Error Set Enable.

8.13.1.41 ee_sd_en_clr_ev1 Register (offset = 0x4420)

Table 8-313. ee_sd_en_clr_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	WRITE	SD B8 PLL Error Clear Enable.
0	sd_ee_stspll_b4_err	WRITE	SD B4 PLL Error Clear Enable.

8.13.1.42 ee_sd_en_sts_ev0 Register (offset = 0x4424)

Table 8-314. ee_sd_en_sts_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	READ	SD B8 PLL Error Enabled.
0	sd_ee_stspll_b4_err	READ	SD B4 PLL Error Enabled.

8.13.1.43 ee_sd_en_sts_ev1 Register (offset = 0x4428)

Table 8-315. ee_sd_en_sts_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	sd_ee_stspll_b8_err	READ	SD B8 PLL Error Enabled.
0	sd_ee_stspll_b4_err	READ	SD B4 PLL Error Enabled.

8.13.1.44 ee_vc_irs Register (offset = 0x4500)

Table 8-316. ee_vc_irs Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	READ	(**Error**) shows VBUS data error

8.13.1.45 ee_vc_irs_set Register (offset = 0x4504)

Table 8-317. ee_vc_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	WRITE	EE VC VBUS Error Set.

8.13.1.46 ee_vc_irs_clr Register (offset = 0x4508)

Table 8-318. ee_vc_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	WRITE	EE VC VBUS Error Clear

8.13.1.47 ee_vc_en_ev0 Register (offset = 0x450C)

Table 8-319. ee_vc_en_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	READ	EE VC VBUS Error Enable.

8.13.1.48 ee_vc_en_set_ev0 Register (offset = 0x4510)

Table 8-320. ee_vc_en_set_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	WRITE	EE VC VBUS Error Set Enable.

8.13.1.49 ee_vc_en_clr_ev0 Register (offset = 0x4514)

Table 8-321. ee_vc_en_clr_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	WRITE	EE VC VBUS Error Clear Enable.

8.13.1.50 ee_vc_en_ev1 Register (offset = 0x4518)

Table 8-322. ee_vc_en_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	READ	EE VC VBUS Error Enable.

8.13.1.51 ee_vc_en_set_ev1 Register (offset = 0x451C)

Table 8-323. ee_vc_en_set_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	WRITE	EE VC VBUS Error Set Enable.

8.13.1.52 ee_vc_en_clr_ev1 Register (offset = 0x4520)

Table 8-324. ee_vc_en_clr_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	WRITE	EE VC VBUS Error Clear Enable.

8.13.1.53 ee_vc_en_sts_ev0 Register (offset = 0x4524)

Table 8-325. ee_vc_en_sts_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	READ	EE VC VBUS Error Enabled.

8.13.1.54 ee_vc_en_sts_ev1 Register (offset = 0x4528)

Table 8-326. ee_vc_en_sts_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	vc_ee_vbus_err	READ	EE VC VBUS Error Enabled

8.13.1.55 EE_AIF2_RUN_STS Register (offset = 0x4600)

Table 8-327. EE_AIF2_RUN_STS Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	aif2_global_run	READ	This bit indicates the overall state of the AIF2. 0 = AIF2 has not been determined a stable 1 = AIF2 has been determined a stable
0	aif2_phy_run	READ	This bit indicates the state of the AIF2 Phy section. 0 = AIF2 Phy has not been determined a stable 1 = AIF2 Phy has been determined a stable

8.13.1.56 EE_AIF2_RUN_CTL Register (offset = 0x4604)

Table 8-328. EE_AIF2_RUN_CTL Register Field Descriptions

Bits	Field Name	Type	Description
31-2	RSVD	NOT_ACCESSIBLE	Reserved.
1	aif2_global_run	WRITE	This bit indicates the overall state of the AIF2. 0 = AIF2 has not been determined a stable 1 = AIF2 has been determined a stable
0	aif2_phy_run	WRITE	This bit indicates the state of the AIF2 Phy section. 0 = AIF2 Phy has not been determined a stable 1 = AIF2 Phy has been determined a stable

8.13.1.57 ee_err_alrm_orgn Register (offset = 0x4700)

Table 8-329. ee_err_alrm_orgn Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	vc_en_sts	READ	VC_EN_STS register indication.
16	cd_en_sts	READ	CD_EN_STS register indication.
15	ad_en_sts	READ	AD_EN_STS register indication.
14	db_en_sts	READ	DB_EN_STS register indication.
13	sd_en_sts	READ	SD_EN_STS register indication.
12	at_en_sts	READ	AT_EN_STS register indication.
11	lk_en_sts_b5	READ	EE_LK_EN_STS_B5 register indication.
10	lk_en_sts_a5	READ	EE_LK_EN_STS_A5 register indication.
9	lk_en_sts_b4	READ	EE_LK_EN_STS_B4 register indication.
8	lk_en_sts_a4	READ	EE_LK_EN_STS_A4 register indication.
7	lk_en_sts_b3	READ	EE_LK_EN_STS_B3 register indication.
6	lk_en_sts_a3	READ	EE_LK_EN_STS_A3 register indication.
5	lk_en_sts_b2	READ	EE_LK_EN_STS_B2 register indication.
4	lk_en_sts_a2	READ	EE_LK_EN_STS_A2 register indication.
3	lk_en_sts_b1	READ	EE_LK_EN_STS_B1 register indication.
2	lk_en_sts_a1	READ	EE_LK_EN_STS_A1 register indication.
1	lk_en_sts_b0	READ	EE_LK_EN_STS_B0 register indication.
0	lk_en_sts_a0	READ	EE_LK_EN_STS_A0 register indication.

8.13.1.58 ee_lk_irs_a[0] Register (offset = 0x40000)

Table 8-330. ee_lk_irs_a[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	READ	(**Informational or Error**), Per link. The TM FIFO for link does not have data to transmit. This bit is active when the TM FIFO does not have data in it
24	tm_ee_frm_misalign_err	READ	(**Informational or Error**), Per Link. The TM state machine detected a mis-aligned data frame on link. This bit is active on any clock cycle that the frame indication for the TM from the AT block does not line up with the frame indication in the data path. A misalignment will also occur if a frame indication from the AT block happens but the TM Fifo is empty
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	READ	Per Link, CPRI only (**Error**): Indicates that the RX FSM is in the Loss Of Signal state that is, ST0 (as defined by CPRI)
16	rm_ee_hfnsync_state_err	READ	Per Link, CPRI only (**Error**): Indicates RX FSM in the hyperframe state that is, state ST3. (as defined by CPRI)
15	rm_ee_lof_err	READ	Per Link, CPRI only (**Error**): Indicates that the RX FSM is in the Loss Of Frame state that is, ST0 or St1 (as defined by CPRI)
14	rm_ee_los_err	READ	Per Link, CPRI only (**Error**): Indicates that the RX FSM is in the Loss Of Signal state that is, ST0 (as defined by CPRI)
13	rm_ee_rcvd_sdi_err	READ	Per Link, CPRI only (**Error**): Indicates SDI error.
12	rm_ee_rcvd_rai_err	READ	Per Link, CPRI only (**Error**): Indicates RAI error.
11	rm_ee_rcvd_lof_err	READ	Per Link, CPRI only (**Error**): Indicates a received, L1 Inband Loss Of Frame
10	rm_ee_rcvd_los_err	READ	Per Link, CPRI only (**Error**): Indicates a received, L1 Inband Loss Of Signal event (Z.130.0, bit3 as defined by CPRI)
9	rm_ee_rx_fifo_ovf_err	READ	Per Link (**Error**): Indicates that the RX FIFO has overflowed.
8	rm_ee_loc_det_err	READ	Per Link (**Error**): Indicates that the loss of clock watch dog timer has timed out that is, SerDes clock was not detected within the specified window.
7	rm_ee_k30p7_det_err	READ	Per Link, OBSAI Only (**Error**): Indicates that a k30.7 error character was received.
6	rm_ee_missing_k28p7_err	READ	Per Link, OBSAI Only (**Error**): Indicates that a k28.7 character was not received when it was supposed to have been received.
5	rm_ee_missing_k28p5_err	READ	Per Link (**Error**): Indicates that a k28.5 character was not received when it was supposed to have been received.
4	rm_ee_block_bndry_det_err	READ	Per Link (**Info**): Indicates that a block boundary was detected. CPRI: K28.5 character & HFN!= 149 OBSAI: K28.5 character.
3	rm_ee_frame_bndry_det_err	READ	Per Link (**Info**): Indicates that a frame boundary was detected.
2	rm_ee_lcv_det_err	READ	Per Link (**Error**): lcv detection error
1	rm_ee_num_los_det_err	READ	Per Link (**Error**): los detection error
0	rm_ee_sync_status_change_err	READ	Per Link (**Information**): Indicates that the RX state machine changed state

ee_lk_irs_a[1]	**Access = READ**	**Address [0x40080]**
ee_lk_irs_a[2]	**Access = READ**	**Address [0x40100]**
ee_lk_irs_a[3]	**Access = READ**	**Address [0x40180]**
ee_lk_irs_a[4]	**Access = READ**	**Address [0x40200]**
ee_lk_irs_a[5]	**Access = READ**	**Address [0x40280]**

8.13.1.59 ee_lk_irs_set_a[0] Register (offset = 0x40004)

Table 8-331. ee_lk_irs_set_a[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	WRITE	TM FIFO Starve Error Set.
24	tm_ee_frm_misalign_err	WRITE	TM Frame Misalign Error Set.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	WRITE	RM LOF State Error Set.
16	rm_ee_hfnsync_state_err	WRITE	RM Hyperframe State Error Set.
15	rm_ee_lof_err	WRITE	RM LOF Error Set.
14	rm_ee_los_err	WRITE	RM LOS Error Set.
13	rm_ee_rcvd_sdi_err	WRITE	RM Received SDI Error Set.
12	rm_ee_rcvd_rai_err	WRITE	RM Received RAI Error Set.
11	rm_ee_rcvd_lof_err	WRITE	RM Received LOF Error Set.
10	rm_ee_rcvd_los_err	WRITE	RM Received LOS Error Set.
9	rm_ee_rx_fifo_ovf_err	WRITE	RM RX FIFO Overflow Error Set.
8	rm_ee_loc_det_err	WRITE	RM LOC Detect Error Set.
7	rm_ee_k30p7_det_err	WRITE	RM K30p7 Detect Error Set.
6	rm_ee_missing_k28p7_err	WRITE	RM Missing K28p7 Error Set.
5	rm_ee_missing_k28p5_err	WRITE	RM Missing K28p5 Error Set.
4	rm_ee_block_bndry_det_err	WRITE	RM Block Boundary Detect Error Set.
3	rm_ee_frame_bndry_det_err	WRITE	RM Frame Boundary Detect Error Set.
2	rm_ee_lcv_det_err	WRITE	RM LCV Detect Error Set.
1	rm_ee_num_los_det_err	WRITE	RM Num Loss Detect Error Set.
0	rm_ee_sync_status_change_err	WRITE	RM Sync Status Change Error Set.

ee_lk_irs_set_a[1]	**Access = READ**	**Address [0x40084]**
ee_lk_irs_set_a[2]	**Access = READ**	**Address [0x40104]**
ee_lk_irs_set_a[3]	**Access = READ**	**Address [0x40184]**
ee_lk_irs_set_a[4]	**Access = READ**	**Address [0x40204]**
ee_lk_irs_set_a[5]	**Access = READ**	**Address [0x40284]**

8.13.1.60 ee_lk_irs_clr_a[0] Register (offset = 0x40008)

Table 8-332. ee_lk_irs_clr_a[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	WRITE	TM FIFO Starve Error Clear.
24	tm_ee_frm_misalign_err	WRITE	TM Frame Misalign Error Clear.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	WRITE	RM LOF State Error Clear.
16	rm_ee_hfnsync_state_err	WRITE	RM Hyperframe State Error Clear.
15	rm_ee_lof_err	WRITE	RM LOF Error Clear.
14	rm_ee_los_err	WRITE	RM LOS Error Clear.
13	rm_ee_rcvd_sdi_err	WRITE	RM Received SDI Error Clear.
12	rm_ee_rcvd_rai_err	WRITE	RM Received RAI Error Clear.
11	rm_ee_rcvd_lof_err	WRITE	RM Received LOF Error Clear.
10	rm_ee_rcvd_los_err	WRITE	RM Received LOS Error Clear.
9	rm_ee_rx_fifo_ovf_err	WRITE	RM RX FIFO Overflow Error Clear.
8	rm_ee_loc_det_err	WRITE	RM LOC Detect Error Clear.
7	rm_ee_k30p7_det_err	WRITE	RM K30p7 Detect Error Clear.
6	rm_ee_missing_k28p7_err	WRITE	RM Missing K28p7 Error Clear.
5	rm_ee_missing_k28p5_err	WRITE	RM Missing K28p5 Error Clear.
4	rm_ee_block_bndry_det_err	WRITE	RM Block Boundary Detect Error Clear.
3	rm_ee_frame_bndry_det_err	WRITE	RM Frame Boundary Detect Error Clear.
2	rm_ee_lcv_det_err	WRITE	RM LCV Detect Error Clear.
1	rm_ee_num_los_det_err	WRITE	RM Num Loss Detect Error Clear.
0	rm_ee_sync_status_change_err	WRITE	RM Sync Status Change Error Clear.

ee_lk_irs_clr_a[1]	**Access = READ**	**Address [0x40088]**
ee_lk_irs_clr_a[2]	**Access = READ**	**Address [0x40108]**
ee_lk_irs_clr_a[3]	**Access = READ**	**Address [0x40188]**
ee_lk_irs_clr_a[4]	**Access = READ**	**Address [0x40208]**
ee_lk_irs_clr_a[5]	**Access = READ**	**Address [0x40288]**

8.13.1.61 ee_lk_en_a_ev0[0] Register (offset = 0x4000C)

Table 8-333. ee_lk_en_a_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	READ	TM FIFO Starve Error Enable.
24	tm_ee_frm_misalign_err	READ	TM Frame Misalign Error Enable.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	READ	RM LOF State Error Enable.
16	rm_ee_hfnsync_state_err	READ	RM Hyperframe State Error Enable.
15	rm_ee_lof_err	READ	RM LOF Error Enable.
14	rm_ee_los_err	READ	RM LOS Error Enable.
13	rm_ee_rcvd_sdi_err	READ	RM Received SDI Error Enable.
12	rm_ee_rcvd_rai_err	READ	RM Received RAI Error Enable.
11	rm_ee_rcvd_lof_err	READ	RM Received LOF Error Enable.
10	rm_ee_rcvd_los_err	READ	RM Received LOS Error Enable.
9	rm_ee_rx_fifo_ovf_err	READ	RM RX FIFO Overflow Error Enable.
8	rm_ee_loc_det_err	READ	RM LOC Detect Error Enable.
7	rm_ee_k30p7_det_err	READ	RM K30p7 Detect Error Enable.
6	rm_ee_missing_k28p7_err	READ	RM Missing K28p7 Error Enable.
5	rm_ee_missing_k28p5_err	READ	RM Missing K28p5 Error Enable.
4	rm_ee_block_bndry_det_err	READ	RM Block Boundary Detect Error Enable.
3	rm_ee_frame_bndry_det_err	READ	RM Frame Boundary Detect Error Enable.
2	rm_ee_lcv_det_err	READ	RM LCV Detect Error Enable.
1	rm_ee_num_los_det_err	READ	RM Num Loss Detect ErroSet.
0	rm_ee_sync_status_change_err	READ	RM Sync Status Change Error Enable.

ee_lk_en_a_ev0[1]	**Access = READ**	**Address [0x4008C]**
ee_lk_en_a_ev0[2]	**Access = READ**	**Address [0x4010C]**
ee_lk_en_a_ev0[3]	**Access = READ**	**Address [0x4018C]**
ee_lk_en_a_ev0[4]	**Access = READ**	**Address [0x4020C]**
ee_lk_en_a_ev0[5]	**Access = READ**	**Address [0x4028C]**

8.13.1.62 ee_lk_en_a_set_ev0[0] Register (offset = 0x40010)

Table 8-334. ee_lk_en_a_set_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	WRITE	TM FIFO Starve Error Enable Set.
24	tm_ee_frm_misalign_err	WRITE	TM Frame Misalign Error Enable Set.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	WRITE	RM LOF State Error Enable Set.
16	rm_ee_hfnsync_state_err	WRITE	RM Hyperframe State Error Enable Set.
15	rm_ee_lof_err	WRITE	RM LOF Error Enable Set.
14	rm_ee_los_err	WRITE	RM LOS Error Enable Set.
13	rm_ee_rcvd_sdi_err	WRITE	RM Received SDI Error Enable Set.
12	rm_ee_rcvd_rai_err	WRITE	RM Received RAI Error Enable Set.
11	rm_ee_rcvd_lof_err	WRITE	RM Received LOF Error Enable Set.
10	rm_ee_rcvd_los_err	WRITE	RM Received LOS Error Enable Set.
9	rm_ee_rx_fifo_ovf_err	WRITE	RM RX FIFO Overflow Error Enable Set.
8	rm_ee_loc_det_err	WRITE	RM LOC Detect Error Enable Set.
7	rm_ee_k30p7_det_err	WRITE	RM K30p7 Detect Error Enable Set.
6	rm_ee_missing_k28p7_err	WRITE	RM Missing K28p7 Error Enable Set.
5	rm_ee_missing_k28p5_err	WRITE	RM Missing K28p5 Error Enable Set.
4	rm_ee_block_bndry_det_err	WRITE	RM Block Boundary Detect Error Enable Set.
3	rm_ee_frame_bndry_det_err	WRITE	RM Frame Boundary Detect Error Enable Set.
2	rm_ee_lcv_det_err	WRITE	RM LCV Detect Error Enable Set.
1	rm_ee_num_los_det_err	WRITE	RM Num Loss Detect ErroSet.
0	rm_ee_sync_status_change_err	WRITE	RM Sync Status Change Error Enable Set.

ee_lk_en_a_set_ev0[1]	**Access = READ**	**Address [0x40090]**
ee_lk_en_a_set_ev0[2]	**Access = READ**	**Address [0x40110]**
ee_lk_en_a_set_ev0[3]	**Access = READ**	**Address [0x40190]**
ee_lk_en_a_set_ev0[4]	**Access = READ**	**Address [0x40210]**
ee_lk_en_a_set_ev0[5]	**Access = READ**	**Address [0x40290]**

8.13.1.63 ee_lk_en_a_clr_ev0[0] Register (offset = 0x40014)

Table 8-335. ee_lk_en_a_clr_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	WRITE	TM FIFO Starve Error Enable Clear.
24	tm_ee_frm_misalign_err	WRITE	TM Frame Misalign Error Enable Clear.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	WRITE	RM LOF State Error Enable Clear.
16	rm_ee_hfnsync_state_err	WRITE	RM Hyperframe State Error Enable Clear.
15	rm_ee_lof_err	WRITE	RM LOF Error Enable Clear.
14	rm_ee_los_err	WRITE	RM LOS Error Enable Clear.
13	rm_ee_rcvd_sdi_err	WRITE	RM Received SDI Error Enable Clear.
12	rm_ee_rcvd_rai_err	WRITE	RM Received RAI Error Enable Clear.
11	rm_ee_rcvd_lof_err	WRITE	RM Received LOF Error Enable Clear.
10	rm_ee_rcvd_los_err	WRITE	RM Received LOS Error Enable Clear.
9	rm_ee_rx_fifo_ovf_err	WRITE	RM RX FIFO Overflow Error Enable Clear.
8	rm_ee_loc_det_err	WRITE	RM LOC Detect Error Enable Clear.
7	rm_ee_k30p7_det_err	WRITE	RM K30p7 Detect Error Enable Clear.
6	rm_ee_missing_k28p7_err	WRITE	RM Missing K28p7 Error Enable Clear.
5	rm_ee_missing_k28p5_err	WRITE	RM Missing K28p5 Error Enable Clear.
4	rm_ee_block_bndry_det_err	WRITE	RM Block Boundary Detect Error Enable Clear.
3	rm_ee_frame_bndry_det_err	WRITE	RM Frame Boundary Detect Error Enable Clear.
2	rm_ee_lcv_det_err	WRITE	RM LCV Detect Error Enable Clear.
1	rm_ee_num_los_det_err	WRITE	RM Num Loss Detect ErroSet.
0	rm_ee_sync_status_change_err	WRITE	RM Sync Status Change Error Enable Clear.

ee_lk_en_a_clr_ev0[1]	**Access = READ**	**Address [0x40094]**
ee_lk_en_a_clr_ev0[2]	**Access = READ**	**Address [0x40114]**
ee_lk_en_a_clr_ev0[3]	**Access = READ**	**Address [0x40194]**
ee_lk_en_a_clr_ev0[4]	**Access = READ**	**Address [0x40214]**
ee_lk_en_a_clr_ev0[5]	**Access = READ**	**Address [0x40294]**

8.13.1.64 ee_lk_en_a_ev1[0] Register (offset = 0x40018)

Table 8-336. ee_lk_en_a_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	READ	TM FIFO Starve Error Enable.
24	tm_ee_frm_misalign_err	READ	TM Frame Misalign Error Enable.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	READ	RM LOF State Error Enable.
16	rm_ee_hfnsync_state_err	READ	RM Hyperframe State Error Enable.
15	rm_ee_lof_err	READ	RM LOF Error Enable.
14	rm_ee_los_err	READ	RM LOS Error Enable.
13	rm_ee_rcvd_sdi_err	READ	RM Received SDI Error Enable.
12	rm_ee_rcvd_rai_err	READ	RM Received RAI Error Enable.
11	rm_ee_rcvd_lof_err	READ	RM Received LOF Error Enable.
10	rm_ee_rcvd_los_err	READ	RM Received LOS Error Enable.
9	rm_ee_rx_fifo_ovf_err	READ	RM RX FIFO Overflow Error Enable.
8	rm_ee_loc_det_err	READ	RM LOC Detect Error Enable.
7	rm_ee_k30p7_det_err	READ	RM K30p7 Detect Error Enable.
6	rm_ee_missing_k28p7_err	READ	RM Missing K28p7 Error Enable.
5	rm_ee_missing_k28p5_err	READ	RM Missing K28p5 Error Enable.
4	rm_ee_block_bndry_det_err	READ	RM Block Boundary Detect Error Enable.
3	rm_ee_frame_bndry_det_err	READ	RM Frame Boundary Detect Error Enable.
2	rm_ee_lcv_det_err	READ	RM LCV Detect Error Enable.
1	rm_ee_num_los_det_err	READ	RM Num Loss Detect ErroSet.
0	rm_ee_sync_status_change_err	READ	RM Sync Status Change Error Enable.

ee_lk_en_a_ev1[1]	**Access = READ**	**Address [0x40098]**
ee_lk_en_a_ev1[2]	**Access = READ**	**Address [0x40118]**
ee_lk_en_a_ev1[3]	**Access = READ**	**Address [0x40198]**
ee_lk_en_a_ev1[4]	**Access = READ**	**Address [0x40218]**
ee_lk_en_a_ev1[5]	**Access = READ**	**Address [0x40298]**

8.13.1.65 ee_lk_en_a_set_ev1[0] Register (offset = 0x4001C)

Table 8-337. ee_lk_en_a_set_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	WRITE	TM FIFO Starve Error Enable Set.
24	tm_ee_frm_misalign_err	WRITE	TM Frame Misalign Error Enable Set.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	WRITE	RM LOF State Error Enable Set.
16	rm_ee_hfnsync_state_err	WRITE	RM Hyperframe State Error Enable Set.
15	rm_ee_lof_err	WRITE	RM LOF Error Enable Set.
14	rm_ee_los_err	WRITE	RM LOS Error Enable Set.
13	rm_ee_rcvd_sdi_err	WRITE	RM Received SDI Error Enable Set.
12	rm_ee_rcvd_rai_err	WRITE	RM Received RAI Error Enable Set.
11	rm_ee_rcvd_lof_err	WRITE	RM Received LOF Error Enable Set.
10	rm_ee_rcvd_los_err	WRITE	RM Received LOS Error Enable Set.
9	rm_ee_rx_fifo_ovf_err	WRITE	RM RX FIFO Overflow Error Enable Set.
8	rm_ee_loc_det_err	WRITE	RM LOC Detect Error Enable Set.
7	rm_ee_k30p7_det_err	WRITE	RM K30p7 Detect Error Enable Set.
6	rm_ee_missing_k28p7_err	WRITE	RM Missing K28p7 Error Enable Set.
5	rm_ee_missing_k28p5_err	WRITE	RM Missing K28p5 Error Enable Set.
4	rm_ee_block_bndry_det_err	WRITE	RM Block Boundary Detect Error Enable Set.
3	rm_ee_frame_bndry_det_err	WRITE	RM Frame Boundary Detect Error Enable Set.
2	rm_ee_lcv_det_err	WRITE	RM LCV Detect Error Enable Set.
1	rm_ee_num_los_det_err	WRITE	RM Num Loss Detect ErroSet.
0	rm_ee_sync_status_change_err	WRITE	RM Sync Status Change Error Enable Set.

ee_lk_en_a_set_ev1[1]	**Access = READ**	**Address [0x4009C]**
ee_lk_en_a_set_ev1[2]	**Access = READ**	**Address [0x4011C]**
ee_lk_en_a_set_ev1[3]	**Access = READ**	**Address [0x4019C]**
ee_lk_en_a_set_ev1[4]	**Access = READ**	**Address [0x4021C]**
ee_lk_en_a_set_ev1[5]	**Access = READ**	**Address [0x4029C]**

8.13.1.66 ee_lk_en_a_clr_ev1[0] Register (offset = 0x40020)

Table 8-338. ee_lk_en_a_clr_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	WRITE	TM FIFO Starve Error Enable Clear.
24	tm_ee_frm_misalign_err	WRITE	TM Frame Misalign Error Enable Clear.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	WRITE	RM LOF State Error Enable Clear.
16	rm_ee_hfnsync_state_err	WRITE	RM Hyperframe State Error Enable Clear.
15	rm_ee_lof_err	WRITE	RM LOF Error Enable Clear.
14	rm_ee_los_err	WRITE	RM LOS Error Enable Clear.
13	rm_ee_rcvd_sdi_err	WRITE	RM Received SDI Error Enable Clear.
12	rm_ee_rcvd_rai_err	WRITE	RM Received RAI Error Enable Clear.
11	rm_ee_rcvd_lof_err	WRITE	RM Received LOF Error Enable Clear.
10	rm_ee_rcvd_los_err	WRITE	RM Received LOS Error Enable Clear.
9	rm_ee_rx_fifo_ovf_err	WRITE	RM RX FIFO Overflow Error Enable Clear.
8	rm_ee_loc_det_err	WRITE	RM LOC Detect Error Enable Clear.
7	rm_ee_k30p7_det_err	WRITE	RM K30p7 Detect Error Enable Clear.
6	rm_ee_missing_k28p7_err	WRITE	RM Missing K28p7 Error Enable Clear.
5	rm_ee_missing_k28p5_err	WRITE	RM Missing K28p5 Error Enable Clear.
4	rm_ee_block_bndry_det_err	WRITE	RM Block Boundary Detect Error Enable Clear.
3	rm_ee_frame_bndry_det_err	WRITE	RM Frame Boundary Detect Error Enable Clear.
2	rm_ee_lcv_det_err	WRITE	RM LCV Detect Error Enable Clear.
1	rm_ee_num_los_det_err	WRITE	RM Num Loss Detect ErroSet.
0	rm_ee_sync_status_change_err	WRITE	RM Sync Status Change Error Enable Clear.

ee_lk_en_a_clr_ev1[1]	**Access = READ**	**Address [0x400A0]**
ee_lk_en_a_clr_ev1[2]	**Access = READ**	**Address [0x40120]**
ee_lk_en_a_clr_ev1[3]	**Access = READ**	**Address [0x401A0]**
ee_lk_en_a_clr_ev1[4]	**Access = READ**	**Address [0x40220]**
ee_lk_en_a_clr_ev1[5]	**Access = READ**	**Address [0x402A0]**

8.13.1.67 ee_lk_en_sts_a_ev0[0] Register (offset = 0x40024)

Table 8-339. ee_lk_en_sts_a_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	READ	TM FIFO Starve Error Enabled.
24	tm_ee_frm_misalign_err	READ	TM Frame Misalign Error Enabled.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	READ	RM LOF State Error Enabled.
16	rm_ee_hfnsync_state_err	READ	RM Hyperframe State Error Enabled.
15	rm_ee_lof_err	READ	RM LOF Error
14	rm_ee_los_err	READ	RM LOS Error Enabled.
13	rm_ee_rcvd_sdi_err	READ	RM Received SDI Error Enabled.
12	rm_ee_rcvd_rai_err	READ	RM Received RAI Error Enabled.
11	rm_ee_rcvd_lof_err	READ	RM Received LOF Error Enabled.
10	rm_ee_rcvd_los_err	READ	RM Received LOS Error Enabled.
9	rm_ee_rx_fifo_ovf_err	READ	RM RX FIFO Overflow Error Enabled.
8	rm_ee_loc_det_err	READ	RM LOC Detect Error Enabled.
7	rm_ee_k30p7_det_err	READ	RM K30p7 Detect Error Enabled.
6	rm_ee_missing_k28p7_err	READ	RM Missing K28p7 Error Enabled.
5	rm_ee_missing_k28p5_err	READ	RM Missing K28p5 Error Enabled.
4	rm_ee_block_bndry_det_err	READ	RM Block Boundary Detect Error Enabled.
3	rm_ee_frame_bndry_det_err	READ	RM Frame Boundary Detect Error Enabled.
2	rm_ee_lcv_det_err	READ	RM LCV Detect Error Enabled.
1	rm_ee_num_los_det_err	READ	RM Num Loss Detect Error Enabled.
0	rm_ee_sync_status_change_err	READ	RM Sync Status Change Error Enabled.

ee_lk_en_sts_a_ev0[1]	**Access = READ**	**Address [0x400A4]**
ee_lk_en_sts_a_ev0[2]	**Access = READ**	**Address [0x40124]**
ee_lk_en_sts_a_ev0[3]	**Access = READ**	**Address [0x401A4]**
ee_lk_en_sts_a_ev0[4]	**Access = READ**	**Address [0x40224]**
ee_lk_en_sts_a_ev0[5]	**Access = READ**	**Address [0x402A4]**

8.13.1.68 ee_lk_en_sts_a_ev1[0] Register (offset = 0x40028)

Table 8-340. ee_lk_en_sts_a_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-26	RSVD1	NOT_ACCESSIBLE	Reserved.
25	tm_ee_fifo_starve_err	READ	TM FIFO Starve Error Enabled.
24	tm_ee_frm_misalign_err	READ	TM Frame Misalign Error Enabled.
23-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	rm_ee_lof_state_err	READ	RM LOF State Error Enabled.
16	rm_ee_hfnsync_state_err	READ	RM Hyperframe State Error Enabled.
15	rm_ee_lof_err	READ	RM LOF Error
14	rm_ee_los_err	READ	RM LOS Error Enabled.
13	rm_ee_rcvd_sdi_err	READ	RM Received SDI Error Enabled.
12	rm_ee_rcvd_rai_err	READ	RM Received RAI Error Enabled.
11	rm_ee_rcvd_lof_err	READ	RM Received LOF Error Enabled.
10	rm_ee_rcvd_los_err	READ	RM Received LOS Error Enabled.
9	rm_ee_rx_fifo_ovf_err	READ	RM RX FIFO Overflow Error Enabled.
8	rm_ee_loc_det_err	READ	RM LOC Detect Error Enabled.
7	rm_ee_k30p7_det_err	READ	RM K30p7 Detect Error Enabled.
6	rm_ee_missing_k28p7_err	READ	RM Missing K28p7 Error Enabled.
5	rm_ee_missing_k28p5_err	READ	RM Missing K28p5 Error Enabled.
4	rm_ee_block_bndry_det_err	READ	RM Block Boundary Detect Error Enabled.
3	rm_ee_frame_bndry_det_err	READ	RM Frame Boundary Detect Error Enabled.
2	rm_ee_lcv_det_err	READ	RM LCV Detect Error Enabled.
1	rm_ee_num_los_det_err	READ	RM Num Loss Detect Error Enabled.
0	rm_ee_sync_status_change_err	READ	RM Sync Status Change Error Enabled.

ee_lk_en_sts_a_ev1[1]	**Access = READ**	**Address [0x400A8]**
ee_lk_en_sts_a_ev1[2]	**Access = READ**	**Address [0x40128]**
ee_lk_en_sts_a_ev1[3]	**Access = READ**	**Address [0x401A8]**
ee_lk_en_sts_a_ev1[4]	**Access = READ**	**Address [0x40228]**
ee_lk_en_sts_a_ev1[5]	**Access = READ**	**Address [0x402A8]**

8.13.1.69 ee_lk_irs_b[0] Register (offset = 0x4002C)

Table 8-341. ee_lk_irs_b[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	READ	(**Error**), Per Link. The Retransmit header does not match the transmit header in OBSAI Aggregate mode for Link. While in OBSAI aggregate mode, the RT block expects that the header will match between the transmitted message from the PE and the Retransmitted message received by the RM. If the header does not match, the PE header is assumed to be the one used to transmit out of the TM and this indication is made active.
27	rt_ee_em_err	READ	(**Information**), Per Link. The RT block created an Empty message in OBSAI Aggregate mode for Link. The RT block will replace a message with an empty message in OBSAI Retransmit mode if it detects that a message header is corrupted as indicated by error indication bits supplied by the RM. When this occurs, this bit is made active
26	rt_ee_unfl_err	READ	(**Informational or Error**), Per Link. The RT FIFO has underflowed in Retransmit or Aggregate mode for Link. The RT fifo contains the data that is to be retransmitted from the RM block while the RT is in Retransmit or Aggregate mode. The RT fifo is emptied for the TM to transmit. If the TM expects data and there is not any data for the TM to transmit, this situation defines an underflow condition. When an underflow condition occurs, this bit is made active, and a state machine will require a new frame boundary from the RM to begin buffering data again.
25	rt_ee_ovfl_err	READ	(**Informational or Error**), Per Link. The RT FIFO has overflowed in Retransmit or Aggregate mode for Link. The RT fifo contains the data that is to be retransmitted from the RM block while the RT is in Retransmit or Aggregate mode. The RT fifo is emptied for the TM to transmit. If the TM does not empty the Fifo as fast as the RM fills it, the RT fifo will suffer an overflow condition. When an overflow condition occurs, this bit is made active, and a state machine will require a new frame boundary from the RM to begin buffering data again.
24	rt_ee_frm_err	READ	(**Error**), Per Link. The Retransmit frame no longer matches the Transmit frame in Aggregate mode for Link. The RT checks the frame structure of the data from the RM. If the frame structure received is not consistent with the programmed frame type(OBSAI or CPRI, link_rate), a frame error condition will occur. When a frame error condition occurs, this bit is made active, the RT will flush the RT Fifo so that it is empty, and a state machine will require a new frame boundary from the RM to begin buffering data again
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	READ	Link-by-Link (**Error**), DB did not have packet data for a pkt channel. This error is only for CPRI packets and occurs when an active packet runs out of data in the middle of a packet.
21	pe_ee_rt_if_err	READ	Link-by-Link (**Error**), RT Interface Error
20	pe_ee_db_starve_err	READ	Link-by-Link (**Error**), DB did not have antenna data for a AxC channel. Likely to occur if DMA was late.
19	pe_ee_mf_fifo_underflow_err	READ	Link-by-Link (**Error**), MF FIFO was empty when a read occurred.
18	pe_ee_mf_fifo_overflow_err	READ	Link-by-Link (**Error**), MF FIFO was full when a write occurred.
17	pe_ee_sym_err	READ	Link-by-Link (**Error**), Symbol index in Multicore Navigator protocol specific header did not match for one or more symbol (Multicore Navigator packet).
16	pe_ee_modrule_err	READ	Link-by-Link (**Error**), More than one modulo rule fired in a single clock cycle
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	READ	Link-by-Link (**Information**), If any value is written to the DB. Used as debug indicating PD is processing traffic for a given link.

Table 8-341. ee_lk_irs_b[0] Register Field Descriptions (continued)

Bits	Field Name	Type	Description
6	pd_ee_obsai_frm_err	READ	Link-by-Link (**Error**), Wrap of the PD_Frame Counters did not predict a Radio Frame Boundary consistent with TS=0 falling within the reception timing window. (PD_Frame)
5	pd_ee_sop_err	READ	Link-by-Link (**Error**), Received a second SOP without an EOP in between
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	READ	Link-by-Link (**Error**), Unrecoverable OBSAI Timestamp error for at least 1 AxC of a given link
2	pd_ee_cpri_frame_err	READ	Link-by-Link (**Error**), PD CPRI Frame error. Arriving traffic (and user configuration of DBM) non-multiple of 4 “samples” per CPRI PHY frame. PD will truncate any traffic at the next PHY FB which is not multiple of 4 messages.
1	pd_ee_crc_err	READ	Link-by-Link (**Error**), CRC failure for any packet (with CRC checking) within the given link.
0	pd_ee_eop_err	READ	Link-by-Link (**Error**), Received a second EOP without an SOP in between

ee_lk_irs_b[1]	**Access = READ**	**Address [0x400AC]**
ee_lk_irs_b[2]	**Access = READ**	**Address [0x4012C]**
ee_lk_irs_b[3]	**Access = READ**	**Address [0x401AC]**
ee_lk_irs_b[4]	**Access = READ**	**Address [0x4022C]**
ee_lk_irs_b[5]	**Access = READ**	**Address [0x402AC]**

8.13.1.70 ee_lk_irs_set_b[0] Register (0x40030)

Table 8-342. ee_lk_irs_set_b[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	WRITE	RT HDR Error Set.
27	rt_ee_em_err	WRITE	RT EM Error Set.
26	rt_ee_unfl_err	WRITE	RT Underflow Error Set.
25	rt_ee_ovfl_err	WRITE	RT Overflow Error Set.
24	rt_ee_frm_err	WRITE	RT Frame Error Set.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	WRITE	PE Packet Starvation Error Set.
21	pe_ee_rt_if_err	WRITE	PE RT Interface Error Set.
20	pe_ee_db_starve_err	WRITE	PE DB Starvation Error Set.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Set.
18	pe_ee_mf_fifo_overflow_err	WRITE	PE MF Fifo Overflow Error Set.
17	pe_ee_sym_err	WRITE	PE Symbol Error Set.
16	pe_ee_modrule_err	WRITE	PE Mod Rule Error Set.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	WRITE	PD Write to DB Error Set.
6	pd_ee_obsai_frm_err	WRITE	PD OBSAI Frame Error Set.
5	pd_ee_sop_err	WRITE	PD SOP Error Set.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	WRITE	PD AXC Fail Error Set.
2	pd_ee_cpri_frame_err	WRITE	PD CPRI Frame Error Set.
1	pd_ee_crc_err	WRITE	PD CRC Error Set.
0	pd_ee_eop_err	WRITE	PD EOP Error Set.

ee_lk_irs_set_b[1]	**Access = READ**	**Address [0x400B0]**
ee_lk_irs_set_b[2]	**Access = READ**	**Address [0x40130]**
ee_lk_irs_set_b[3]	**Access = READ**	**Address [0x401B0]**
ee_lk_irs_set_b[4]	**Access = READ**	**Address [0x40230]**
ee_lk_irs_set_b[5]	**Access = READ**	**Address [0x402B0]**

8.13.1.71 ee_lk_irs_clr_b[0] Register (offset = 0x40034)

Table 8-343. ee_lk_irs_clr_b[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	WRITE	RT HDR Error Clear.
27	rt_ee_em_err	WRITE	RT EM Error Clear.
26	rt_ee_unfl_err	WRITE	RT Underflow Error Clear.
25	rt_ee_ovfl_err	WRITE	RT Overflow Error Clear.
24	rt_ee_frm_err	WRITE	RT Frame Error Clear.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	WRITE	PE Packet Starvation Error Clear.
21	pe_ee_rt_if_err	WRITE	PE RT Interface Error Clear.
20	pe_ee_db_starve_err	WRITE	PE DB Starvation Error Clear.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Clear.
18	pe_ee_mf_fifo_overflow_err	WRITE	PE MF Fifo Overflow Error Clear.
17	pe_ee_sym_err	WRITE	PE Symbol Error Clear.
16	pe_ee_modrule_err	WRITE	PE Mod Rule Error Clear.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	WRITE	PD Write to DB Error Clear.
6	pd_ee_obsai_frm_err	WRITE	PD OBSAI Frame Error Clear.
5	pd_ee_sop_err	WRITE	PD SOP Error Clear.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	WRITE	PD AXC Fail Error Clear.
2	pd_ee_cpri_frame_err	WRITE	PD CPRI Frame Error Clear.
1	pd_ee_crc_err	WRITE	PD CRC Error Clear.
0	pd_ee_eop_err	WRITE	PD EOP Error Clear.

ee_lk_irs_clr_b[1]	**Access = READ**	**Address [0x400B4]**
ee_lk_irs_clr_b[2]	**Access = READ**	**Address [0x40134]**
ee_lk_irs_clr_b[3]	**Access = READ**	**Address [0x401B4]**
ee_lk_irs_clr_b[4]	**Access = READ**	**Address [0x40234]**
ee_lk_irs_clr_b[5]	**Access = READ**	**Address [0x402B4]**

8.13.1.72 ee_lk_en_b_ev0[0] Register (offset = 0x40038)

Table 8-344. ee_lk_en_b_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	READ	RT HDR Error Enable.
27	rt_ee_em_err	READ	RT EM Error Enable.
26	rt_ee_unfl_err	READ	RT Underflow Error Enable.
25	rt_ee_ovfl_err	READ	RT Overflow Error Enable.
24	rt_ee_frm_err	READ	RT Frame Error Enable.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	READ	PE Packet Starvation Error Enable.
21	pe_ee_rt_if_err	READ	PE RT Interface Error Enable.
20	pe_ee_db_starve_err	READ	PE DB Starvation Error Enable.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enable.
18	pe_ee_mf_fifo_overflow_err	READ	PE MF Fifo Overflow Error Enable.
17	pe_ee_sym_err	READ	PE Symbol Error Enable.
16	pe_ee_modrule_err	READ	PE Mod Rule Error Enable.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	READ	PD READ to DB Error Enable.
6	pd_ee_obsai_frm_err	READ	PD OBSAI Frame Error Enable.
5	pd_ee_sop_err	READ	PD SOP Error Enable.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	READ	PD AXC Fail Error Enable.
2	pd_ee_cpri_frame_err	READ	PD CPRI Frame Error Enable.
1	pd_ee_crc_err	READ	PD CRC Error Enable.
0	pd_ee_eop_err	READ	PD EOP Error Enable.

ee_lk_en_b_ev0[1]	**Access = READ**	**Address [0x400B8]**
ee_lk_en_b_ev0[2]	**Access = READ**	**Address [0x40138]**
ee_lk_en_b_ev0[3]	**Access = READ**	**Address [0x401B8]**
ee_lk_en_b_ev0[4]	**Access = READ**	**Address [0x40238]**
ee_lk_en_b_ev0[5]	**Access = READ**	**Address [0x402B8]**

8.13.1.73 ee_lk_en_b_set_ev0[0] Register (offset = 0x4003C)

Table 8-345. ee_lk_en_b_set_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	WRITE	RT HDR Error Enable Set.
27	rt_ee_em_err	WRITE	RT EM Error Enable Set.
26	rt_ee_unfl_err	WRITE	RT Underflow Error Enable Set.
25	rt_ee_ovfl_err	WRITE	RT Overflow Error Enable Set.
24	rt_ee_frm_err	WRITE	RT Frame Error Enable Set.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	WRITE	PE Packet Starvation Error Enable Set.
21	pe_ee_rt_if_err	WRITE	PE RT Interface Error Enable Set.
20	pe_ee_db_starve_err	WRITE	PE DB Starvation Error Enable Set.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enable Set.
18	pe_ee_mf_fifo_overflow_err	WRITE	PE MF Fifo Overflow Error Enable Set.
17	pe_ee_sym_err	WRITE	PE Symbol Error Enable Set.
16	pe_ee_modrule_err	WRITE	PE Mod Rule Error Enable Set.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	WRITE	PD Write to DB Error Enable Set.
6	pd_ee_obsai_frm_err	WRITE	PD OBSAI Frame Error Enable Set.
5	pd_ee_sop_err	WRITE	PD SOP Error Enable Set.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	WRITE	PD AXC Fail Error Enable Set.
2	pd_ee_cpri_frame_err	WRITE	PD CPRI Frame Error Enable Set.
1	pd_ee_crc_err	WRITE	PD CRC Error Enable Set.
0	pd_ee_eop_err	WRITE	PD EOP Error Enable Set.

ee_lk_en_b_set_ev0[1]	**Access = READ**	**Address [0x400BC]**
ee_lk_en_b_set_ev0[2]	**Access = READ**	**Address [0x4013C]**
ee_lk_en_b_set_ev0[3]	**Access = READ**	**Address [0x401BC]**
ee_lk_en_b_set_ev0[4]	**Access = READ**	**Address [0x4023C]**
ee_lk_en_b_set_ev0[5]	**Access = READ**	**Address [0x402BC]**

8.13.1.74 ee_lk_en_b_clr_ev0[0] Register (offset = 0x40040)

Table 8-346. ee_lk_en_b_clr_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	WRITE	RT HDR Error Enable Clear.
27	rt_ee_em_err	WRITE	RT EM Error Enable Clear.
26	rt_ee_unfl_err	WRITE	RT Underflow Error Enable Clear.
25	rt_ee_ovfl_err	WRITE	RT Overflow Error Enable Clear.
24	rt_ee_frm_err	WRITE	RT Frame Error Enable Clear.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	WRITE	PE Packet Starvation Error Enable Clear.
21	pe_ee_rt_if_err	WRITE	PE RT Interface Error Enable Clear.
20	pe_ee_db_starve_err	WRITE	PE DB Starvation Error Enable Clear.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enable Clear.
18	pe_ee_mf_fifo_overflow_err	WRITE	PE MF Fifo Overflow Error Enable Clear.
17	pe_ee_sym_err	WRITE	PE Symbol Error Enable Clear
16	pe_ee_modrule_err	WRITE	PE Mod Rule Error Enable Clear.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	WRITE	PD Write to DB Error Enable Clear.
6	pd_ee_obsai_frm_err	WRITE	PD OBSAI Frame Error Enable Clear.
5	pd_ee_sop_err	WRITE	PD SOP Error Enable Clear.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	WRITE	PD AXC Fail Error Enable Clear.
2	pd_ee_cpri_frame_err	WRITE	PD CPRI Frame Error Enable Clear.
1	pd_ee_crc_err	WRITE	PD CRC Error Enable Clear.
0	pd_ee_eop_err	WRITE	PD EOP Error Enable Clear.

ee_lk_en_b_clr_ev0[1]	**Access = READ**	**Address [0x400C0]**
ee_lk_en_b_clr_ev0[2]	**Access = READ**	**Address [0x40140]**
ee_lk_en_b_clr_ev0[3]	**Access = READ**	**Address [0x401C0]**
ee_lk_en_b_clr_ev0[4]	**Access = READ**	**Address [0x40240]**
ee_lk_en_b_clr_ev0[5]	**Access = READ**	**Address [0x402C0]**

8.13.1.75 ee_lk_en_b_ev1[0] Register (offset = 0x40044)

Table 8-347. ee_lk_en_b_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	READ	RT HDR Error Enable.
27	rt_ee_em_err	READ	RT EM Error Enable.
26	rt_ee_unfl_err	READ	RT Underflow Error Enable.
25	rt_ee_ovfl_err	READ	RT Overflow Error Enable.
24	rt_ee_frm_err	READ	RT Frame Error Enable.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	READ	PE Packet Starvation Error Enable.
21	pe_ee_rt_if_err	READ	PE RT Interface Error Enable.
20	pe_ee_db_starve_err	READ	PE DB Starvation Error Enable.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enable.
18	pe_ee_mf_fifo_overflow_err	READ	PE MF Fifo Overflow Error Enable.
17	pe_ee_sym_err	READ	PE Symbol Error Enable.
16	pe_ee_modrule_err	READ	PE Mod Rule Error Enable.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	READ	PD READ to DB Error Enable.
6	pd_ee_obsai_frm_err	READ	PD OBSAI Frame Error Enable.
5	pd_ee_sop_err	READ	PD SOP Error Enable.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	READ	PD AXC Fail Error Enable.
2	pd_ee_cpri_frame_err	READ	PD CPRI Frame Error Enable.
1	pd_ee_crc_err	READ	PD CRC Error Enable.
0	pd_ee_eop_err	READ	PD EOP Error Enable.

ee_lk_en_b_ev1[1]	**Access = READ**	**Address [0x400C4]**
ee_lk_en_b_ev1[2]	**Access = READ**	**Address [0x40144]**
ee_lk_en_b_ev1[3]	**Access = READ**	**Address [0x401C4]**
ee_lk_en_b_ev1[4]	**Access = READ**	**Address [0x40244]**
ee_lk_en_b_ev1[5]	**Access = READ**	**Address [0x402C4]**

8.13.1.76 ee_lk_en_b_set_ev1[0] Register (offset = 0x40048)

Table 8-348. ee_lk_en_b_set_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	WRITE	RT HDR Error Enable Set.
27	rt_ee_em_err	WRITE	RT EM Error Enable Set.
26	rt_ee_unfl_err	WRITE	RT Underflow Error Enable Set.
25	rt_ee_ovfl_err	WRITE	RT Overflow Error Enable Set.
24	rt_ee_frm_err	WRITE	RT Frame Error Enable Set.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	WRITE	PE Packet Starvation Error Enable Set.
21	pe_ee_rt_if_err	WRITE	PE RT Interface Error Enable Set.
20	pe_ee_db_starve_err	WRITE	PE DB Starvation Error Enable Set.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enable Set.
18	pe_ee_mf_fifo_overflow_err	WRITE	PE MF Fifo Overflow Error Enable Set.
17	pe_ee_sym_err	WRITE	PE Symbol Error Enable Set.
16	pe_ee_modrule_err	WRITE	PE Mod Rule Error Enable Set.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	WRITE	PD Write to DB Error Enable Set.
6	pd_ee_obsai_frm_err	WRITE	PD OBSAI Frame Error Enable Set.
5	pd_ee_sop_err	WRITE	PD SOP Error Enable Set.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	WRITE	PD AXC Fail Error Enable Set.
2	pd_ee_cpri_frame_err	WRITE	PD CPRI Frame Error Enable Set.
1	pd_ee_crc_err	WRITE	PD CRC Error Enable Set.
0	pd_ee_eop_err	WRITE	PD EOP Error Enable Set.

ee_lk_en_b_set_ev1[1]	**Access = READ**	**Address [0x400C8]**
ee_lk_en_b_set_ev1[2]	**Access = READ**	**Address [0x40148]**
ee_lk_en_b_set_ev1[3]	**Access = READ**	**Address [0x401C8]**
ee_lk_en_b_set_ev1[4]	**Access = READ**	**Address [0x40248]**
ee_lk_en_b_set_ev1[5]	**Access = READ**	**Address [0x402C8]**

8.13.1.77 ee_lk_en_b_clr_ev1[0] Register (offset = 0x4004C)

Table 8-349. ee_lk_en_b_clr_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	WRITE	RT HDR Error Enable Clear.
27	rt_ee_em_err	WRITE	RT EM Error Enable Clear.
26	rt_ee_unfl_err	WRITE	RT Underflow Error Enable Clear.
25	rt_ee_ovfl_err	WRITE	RT Overflow Error Enable Clear.
24	rt_ee_frm_err	WRITE	RT Frame Error Enable Clear.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	WRITE	PE Packet Starvation Error Enable Clear.
21	pe_ee_rt_if_err	WRITE	PE RT Interface Error Enable Clear.
20	pe_ee_db_starve_err	WRITE	PE DB Starvation Error Enable Clear.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enable Clear.
18	pe_ee_mf_fifo_overflow_err	WRITE	PE MF Fifo Overflow Error Enable Clear.
17	pe_ee_sym_err	WRITE	PE Symbol Error Enable Clear
16	pe_ee_modrule_err	WRITE	PE Mod Rule Error Enable Clear.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	WRITE	PD Write to DB Error Enable Clear.
6	pd_ee_obsai_frm_err	WRITE	PD OBSAI Frame Error Enable Clear.
5	pd_ee_sop_err	WRITE	PD SOP Error Enable Clear.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	WRITE	PD AXC Fail Error Enable Clear.
2	pd_ee_cpri_frame_err	WRITE	PD CPRI Frame Error Enable Clear.
1	pd_ee_crc_err	WRITE	PD CRC Error Enable Clear.
0	pd_ee_eop_err	WRITE	PD EOP Error Enable Clear.

ee_lk_en_b_clr_ev1[1]	**Access = READ**	**Address [0x400CC]**
ee_lk_en_b_clr_ev1[2]	**Access = READ**	**Address [0x4014C]**
ee_lk_en_b_clr_ev1[3]	**Access = READ**	**Address [0x401CC]**
ee_lk_en_b_clr_ev1[4]	**Access = READ**	**Address [0x4024C]**
ee_lk_en_b_clr_ev1[5]	**Access = READ**	**Address [0x402CC]**

8.13.1.78 ee_lk_en_sts_b_ev0[0] Register (offset = 0x40050)

Table 8-350. ee_lk_en_sts_b_ev0[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	READ	RT HDR Error Enabled.
27	rt_ee_em_err	READ	RT EM Error Enabled.
26	rt_ee_unfl_err	READ	RT Underflow Error Enabled.
25	rt_ee_ovfl_err	READ	RT Overflow Error Enabled.
24	rt_ee_frm_err	READ	RT Frame Error Enabled.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	READ	PE Packet Starvation Error Enabled.
21	pe_ee_rt_if_err	READ	PE RT Interface Error Enabled.
20	pe_ee_db_starve_err	READ	PE DB Starvation Error Enabled.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enabled.
18	pe_ee_mf_fifo_overflow_err	READ	PE MF Fifo Overflow Error Enabled.
17	pe_ee_sym_err	READ	PE Symbol Error Enabled.
16	pe_ee_modrule_err	READ	PE Mod Rule Error Enabled.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	READ	PD Write to DB Error Enabled.
6	pd_ee_obsai_frm_err	READ	PD OBSAI Frame Error Enabled.
5	pd_ee_sop_err	READ	PD SOP Error Enabled.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	READ	PD AXC Fail Error Enabled.
2	pd_ee_cpri_frame_err	READ	PD CPRI Frame Error Enabled.
1	pd_ee_crc_err	READ	PD CRC Error Enabled.
0	pd_ee_eop_err	READ	PD EOP Error Enabled.

ee_lk_en_sts_b_ev0[1]	**Access = READ**	**Address [0x400D0]**
ee_lk_en_sts_b_ev0[2]	**Access = READ**	**Address [0x40150]**
ee_lk_en_sts_b_ev0[3]	**Access = READ**	**Address [0x401D0]**
ee_lk_en_sts_b_ev0[4]	**Access = READ**	**Address [0x40250]**
ee_lk_en_sts_b_ev0[5]	**Access = READ**	**Address [0x402D0]**

8.13.1.79 ee_lk_en_sts_b_ev1[0] Register (offset = 0x40054)

Table 8-351. ee_lk_en_sts_b_ev1[0] Register Field Descriptions

Bits	Field Name	Type	Description
31-29	RSVD3	NOT_ACCESSIBLE	Reserved.
28	rt_ee_hdr_err	READ	RT HDR Error Enabled.
27	rt_ee_em_err	READ	RT EM Error Enabled.
26	rt_ee_unfl_err	READ	RT Underflow Error Enabled.
25	rt_ee_ovfl_err	READ	RT Overflow Error Enabled.
24	rt_ee_frm_err	READ	RT Frame Error Enabled.
23	RSVD2	NOT_ACCESSIBLE	Reserved.
22	pe_ee_pkt_starve_err	READ	PE Packet Starvation Error Enabled.
21	pe_ee_rt_if_err	READ	PE RT Interface Error Enabled.
20	pe_ee_db_starve_err	READ	PE DB Starvation Error Enabled.
19	pe_ee_mf_fifo_underflow_err	NOT_ACCESSIBLE	PE MF Fifo Underflow Error Enabled.
18	pe_ee_mf_fifo_overflow_err	READ	PE MF Fifo Overflow Error Enabled.
17	pe_ee_sym_err	READ	PE Symbol Error Enabled.
16	pe_ee_modrule_err	READ	PE Mod Rule Error Enabled.
15-8	RSVD1	NOT_ACCESSIBLE	Reserved.
7	pd_ee_wr2db_err	READ	PD Write to DB Error Enabled.
6	pd_ee_obsai_frm_err	READ	PD OBSAI Frame Error Enabled.
5	pd_ee_sop_err	READ	PD SOP Error Enabled.
4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pd_ee_axc_fail_err	READ	PD AXC Fail Error Enabled.
2	pd_ee_cpri_frame_err	READ	PD CPRI Frame Error Enabled.
1	pd_ee_crc_err	READ	PD CRC Error Enabled.
0	pd_ee_eop_err	READ	PD EOP Error Enabled.

ee_lk_en_sts_b_ev1[1]	**Access = READ**	**Address [0x400D4]**
ee_lk_en_sts_b_ev1[2]	**Access = READ**	**Address [0x40154]**
ee_lk_en_sts_b_ev1[3]	**Access = READ**	**Address [0x401D4]**
ee_lk_en_sts_b_ev1[4]	**Access = READ**	**Address [0x40254]**
ee_lk_en_sts_b_ev1[5]	**Access = READ**	**Address [0x402D4]**

8.13.1.80 ee_at_irs Register (offset = 0x40300)

Table 8-352. ee_at_irs Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	READ	(**Error**) AT RADT sync input is not aligned to the RADT counter frame boundary Error.
16	at_ee_phyt_sync_err	READ	(**Error**) AT PHYT sync input is not aligned to the PHYT counter frame boundary Error.
15	at_ee_pi5_err	READ	(**Error**) AT Link 5 Captured Pi is not within pi window
14	at_ee_pi4_err	READ	(**Error**) AT Link 4 Captured Pi is not within pi window
13	at_ee_pi3_err	READ	(**Error**) AT Link 3 Captured Pi is not within pi window
12	at_ee_pi2_err	READ	(**Error**) AT Link 2 Captured Pi is not within pi window
11	at_ee_pi1_err	READ	(**Error**) AT Link 1 Captured Pi is not within pi window
10	at_ee_pi0_err	READ	(**Error**) AT Link 0 Captured Pi is not within pi window
9	at_ee_rp1_sys_err	READ	(**Error**) AT RADT selected width Frame number does not match the received RP1 SYS frame number Error.
8	at_ee_rp1_rp3_err	READ	(**Error**) AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error.
7	at_ee_rp1_crc_err	READ	(**Error**) AT RP1 Type CRC Error.
6	at_ee_rp1_bit_width_err	READ	(**Error**) AT RP1 Type Bit Width Error.
5	at_ee_rp1_type_rsvd_err	READ	(**Error**) AT RP1 Type Reserved Error.
4	at_ee_rp1_type_spare_err	READ	(**Error**) AT RP1 Type Spare Error.
3	at_ee_rp1_type_unsel_err	READ	(**Error**) AT RP1 Type Unsel Error.
2	at_ee_rp1_type_tod_rcvd_err	READ	(**Information**) AT RP1 Type TOD data Receive
1	at_ee_rp1_type_rp3_rcvd_err	READ	(**Information**) AT RP1 Type RP3 data Receive
0	at_ee_rp1_type_sys_rcvd_err	READ	(**Information**) AT RP1 Type SYS data Receive

8.13.1.81 ee_at_irs_set Register (offset = 0x40304)

Table 8-353. ee_at_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	WRITE	AT RADT sync input is not aligned to the RADT counter frame boundary Error Set.
16	at_ee_phyt_sync_err	WRITE	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Set.
15	at_ee_pi5_err	WRITE	AT Link 5 PI Error Set.
14	at_ee_pi4_err	WRITE	AT Link 4 PI Error Set.
13	at_ee_pi3_err	WRITE	AT Link 3 PI Error Set.
12	at_ee_pi2_err	WRITE	AT Link 2 PI Error Set.
11	at_ee_pi1_err	WRITE	AT Link 1 PI Error Set.
10	at_ee_pi0_err	WRITE	AT Link 0 PI Error Set.
9	at_ee_rp1_sys_err	WRITE	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Set.
8	at_ee_rp1_rp3_err	WRITE	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Set.
7	at_ee_rp1_crc_err	WRITE	AT RP1 Type CRC Error Set.
6	at_ee_rp1_bit_width_err	WRITE	AT RP1 Type Bit Width Error Set.
5	at_ee_rp1_type_rsvd_err	WRITE	AT RP1 Type Reserved Error Set.
4	at_ee_rp1_type_spare_err	WRITE	AT RP1 Type Spare Error Set.
3	at_ee_rp1_type_unsel_err	WRITE	AT RP1 Type Unsel Error Set.
2	at_ee_rp1_type_tod_rcvd_err	WRITE	AT RP1 Type TOD Receive Error Set.
1	at_ee_rp1_type_rp3_rcvd_err	WRITE	AT RP1 Type RP3 Receive Error Set.
0	at_ee_rp1_type_sys_rcvd_err	WRITE	AT RP1 Type SYS Receive Error Set.

8.13.1.82 ee_at_irs_clr Register (offset = 0x40308)

Table 8-354. ee_at_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	WRITE	AT RADT sync input is not aligned to the RADT counter frame boundary Error Clear.
16	at_ee_phyt_sync_err	WRITE	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Clear.
15	at_ee_pi5_err	WRITE	AT Link 5 PI Error Clear.
14	at_ee_pi4_err	WRITE	AT Link 4 PI Error Clear.
13	at_ee_pi3_err	WRITE	AT Link 3 PI Error Clear.
12	at_ee_pi2_err	WRITE	AT Link 2 PI Error Clear.
11	at_ee_pi1_err	WRITE	AT Link 1 PI Error Clear.
10	at_ee_pi0_err	WRITE	AT Link 0 PI Error Clear.
9	at_ee_rp1_sys_err	WRITE	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Clear.
8	at_ee_rp1_rp3_err	WRITE	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Clear.
7	at_ee_rp1_crc_err	WRITE	AT RP1 Type CRC Error Clear.
6	at_ee_rp1_bit_width_err	WRITE	AT RP1 Type Bit Width Error Clear.
5	at_ee_rp1_type_rsvd_err	WRITE	AT RP1 Type Reserved Error Clear.
4	at_ee_rp1_type_spare_err	WRITE	AT RP1 Type Spare Error Clear.
3	at_ee_rp1_type_unsel_err	WRITE	AT RP1 Type Unsel Error Clear.
2	at_ee_rp1_type_tod_rcvd_err	WRITE	AT RP1 Type TOD Receive Error Clear.
1	at_ee_rp1_type_rp3_rcvd_err	WRITE	AT RP1 Type RP3 Receive Error Clear.
0	at_ee_rp1_type_sys_rcvd_err	WRITE	AT RP1 Type SYS Receive Error Clear.

8.13.1.83 ee_at_en_ev0 Register (offset = 0x4030C)

Table 8-355. ee_at_en_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	READ	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enable.
16	at_ee_phyt_sync_err	READ	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enable.
15	at_ee_pi5_err	READ	AT Link 5 PI Error Enable.
14	at_ee_pi4_err	READ	AT Link 4 PI Error Enable.
13	at_ee_pi3_err	READ	AT Link 3 PI Error Enable.
12	at_ee_pi2_err	READ	AT Link 2 PI Error Enable.
11	at_ee_pi1_err	READ	AT Link 1 PI Error Enable.
10	at_ee_pi0_err	READ	AT Link 0 PI Error Enable.
9	at_ee_rp1_sys_err	READ	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enable.
8	at_ee_rp1_rp3_err	READ	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enable.
7	at_ee_rp1_crc_err	READ	AT RP1 Type CRC Error Enable.
6	at_ee_rp1_bit_width_err	READ	AT RP1 Type Bit Width Error Enable.
5	at_ee_rp1_type_rsvd_err	READ	AT RP1 Type Reserved Error Enable.
4	at_ee_rp1_type_spare_err	READ	AT RP1 Type Spare Error Enable.
3	at_ee_rp1_type_unsel_err	READ	AT RP1 Type Unsel Error Enable.
2	at_ee_rp1_type_tod_rcvd_err	READ	AT RP1 Type TOD Receive Error Enable.
1	at_ee_rp1_type_rp3_rcvd_err	READ	AT RP1 Type RP3 Receive Error Enable.
0	at_ee_rp1_type_sys_rcvd_err	READ	AT RP1 Type SYS Receive Error Enable.

8.13.1.84 ee_at_en_set_ev0 Register (offset = 0x40310)

Table 8-356. ee_at_en_set_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	WRITE	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enable Set.
16	at_ee_phyt_sync_err	WRITE	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enable Set.
15	at_ee_pi5_err	WRITE	AT Link 5 PI Error Enable Set.
14	at_ee_pi4_err	WRITE	AT Link 4 PI Error Enable Set.
13	at_ee_pi3_err	WRITE	AT Link 3 PI Error Enable Set.
12	at_ee_pi2_err	WRITE	AT Link 2 PI Error Enable Set.
11	at_ee_pi1_err	WRITE	AT Link 1 PI Error Enable Set.
10	at_ee_pi0_err	WRITE	AT Link 0 PI Error Enable Set.
9	at_ee_rp1_sys_err	WRITE	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enable Set.
8	at_ee_rp1_rp3_err	WRITE	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enable Set.
7	at_ee_rp1_crc_err	WRITE	AT RP1 Type CRC Error Enable Set.
6	at_ee_rp1_bit_width_err	WRITE	AT RP1 Type Bit Width Error Enable Set.
5	at_ee_rp1_type_rsvd_err	WRITE	AT RP1 Type Reserved Error Enable Set.
4	at_ee_rp1_type_spare_err	WRITE	AT RP1 Type Spare Error Enable Set.
3	at_ee_rp1_type_unsel_err	WRITE	AT RP1 Type Unsel Error Enable Set.
2	at_ee_rp1_type_tod_rcvd_err	WRITE	AT RP1 Type TOD Receive Error Enable Set.
1	at_ee_rp1_type_rp3_rcvd_err	WRITE	AT RP1 Type RP3 Receive Error Enable Set.
0	at_ee_rp1_type_sys_rcvd_err	WRITE	AT RP1 Type SYS Receive Error Enable Set.

8.13.1.85 ee_at_en_clr_ev0 Register (offset = 0x40314)

Table 8-357. ee_at_en_clr_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	WRITE	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enable Clear.
16	at_ee_phyt_sync_err	WRITE	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enable Clear.
15	at_ee_pi5_err	WRITE	AT Link 5 PI Error Enable Clear.
14	at_ee_pi4_err	WRITE	AT Link 4 PI Error Enable Clear.
13	at_ee_pi3_err	WRITE	AT Link 3 PI Error Enable Clear.
12	at_ee_pi2_err	WRITE	AT Link 2 PI Error Enable Clear.
11	at_ee_pi1_err	WRITE	AT Link 1 PI Error Enable Clear.
10	at_ee_pi0_err	WRITE	AT Link 0 PI Error Enable Clear.
9	at_ee_rp1_sys_err	WRITE	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enable Clear.
8	at_ee_rp1_rp3_err	WRITE	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enable Clear.
7	at_ee_rp1_crc_err	WRITE	AT RP1 Type CRC Error Enable Clear.
6	at_ee_rp1_bit_width_err	WRITE	AT RP1 Type Bit Width Error Enable Clear.
5	at_ee_rp1_type_rsvd_err	WRITE	AT RP1 Type Reserved Error.
4	at_ee_rp1_type_spare_err	WRITE	AT RP1 Type Spare Error.
3	at_ee_rp1_type_unsel_err	WRITE	AT RP1 Type Unsel Error Enable Clear.
2	at_ee_rp1_type_tod_rcvd_err	WRITE	AT RP1 Type TOD Receive Error Enable Clear.
1	at_ee_rp1_type_rp3_rcvd_err	WRITE	AT RP1 Type RP3 Receive Error Enable Clear.
0	at_ee_rp1_type_sys_rcvd_err	WRITE	AT RP1 Type SYS Receive Error Enable Clear.

8.13.1.86 ee_at_en_ev1 Register (offset = 0x40318)

Table 8-358. ee_at_en_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	READ	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enable.
16	at_ee_phyt_sync_err	READ	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enable.
15	at_ee_pi5_err	READ	AT Link 5 PI Error Enable.
14	at_ee_pi4_err	READ	AT Link 4 PI Error Enable.
13	at_ee_pi3_err	READ	AT Link 3 PI Error Enable.
12	at_ee_pi2_err	READ	AT Link 2 PI Error Enable.
11	at_ee_pi1_err	READ	AT Link 1 PI Error Enable.
10	at_ee_pi0_err	READ	AT Link 0 PI Error Enable.
9	at_ee_rp1_sys_err	READ	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enable.
8	at_ee_rp1_rp3_err	READ	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enable.
7	at_ee_rp1_crc_err	READ	AT RP1 Type CRC Error Enable.
6	at_ee_rp1_bit_width_err	READ	AT RP1 Type Bit Width Error Enable.
5	at_ee_rp1_type_rsvd_err	READ	AT RP1 Type Reserved Error Enable.
4	at_ee_rp1_type_spare_err	READ	AT RP1 Type Spare Error Enable.
3	at_ee_rp1_type_unsel_err	READ	AT RP1 Type Unsel Error Enable.
2	at_ee_rp1_type_tod_rcvd_err	READ	AT RP1 Type TOD Receive Error Enable.
1	at_ee_rp1_type_rp3_rcvd_err	READ	AT RP1 Type RP3 Receive Error Enable.
0	at_ee_rp1_type_sys_rcvd_err	READ	AT RP1 Type SYS Receive Error Enable.

8.13.1.87 ee_at_en_set_ev1 Register (offset = 0x4031C)

Table 8-359. ee_at_en_set_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	WRITE	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enable Set.
16	at_ee_phyt_sync_err	WRITE	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enable Set.
15	at_ee_pi5_err	WRITE	AT Link 5 PI Error Enable Set.
14	at_ee_pi4_err	WRITE	AT Link 4 PI Error Enable Set.
13	at_ee_pi3_err	WRITE	AT Link 3 PI Error Enable Set.
12	at_ee_pi2_err	WRITE	AT Link 2 PI Error Enable Set.
11	at_ee_pi1_err	WRITE	AT Link 1 PI Error Enable Set.
10	at_ee_pi0_err	WRITE	AT Link 0 PI Error Enable Set.
9	at_ee_rp1_sys_err	WRITE	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enable Set.
8	at_ee_rp1_rp3_err	WRITE	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enable Set.
7	at_ee_rp1_crc_err	WRITE	AT RP1 Type CRC Error Enable Set.
6	at_ee_rp1_bit_width_err	WRITE	AT RP1 Type Bit Width Error Enable Set.
5	at_ee_rp1_type_rsvd_err	WRITE	AT RP1 Type Reserved Error Enable Set.
4	at_ee_rp1_type_spare_err	WRITE	AT RP1 Type Spare Error Enable Set.
3	at_ee_rp1_type_unsel_err	WRITE	AT RP1 Type Unsel Error Enable Set.
2	at_ee_rp1_type_tod_rcvd_err	WRITE	AT RP1 Type TOD Receive Error Enable Set.
1	at_ee_rp1_type_rp3_rcvd_err	WRITE	AT RP1 Type RP3 Receive Error Enable Set.
0	at_ee_rp1_type_sys_rcvd_err	WRITE	AT RP1 Type SYS Receive Error Enable Set.

8.13.1.88 ee_at_en_clr_ev1 Register (offset = 0x40320)

Table 8-360. ee_at_en_clr_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	WRITE	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enable Clear.
16	at_ee_phyt_sync_err	WRITE	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enable Clear.
15	at_ee_pi5_err	WRITE	AT Link 5 PI Error Enable Clear.
14	at_ee_pi4_err	WRITE	AT Link 4 PI Error Enable Clear.
13	at_ee_pi3_err	WRITE	AT Link 3 PI Error Enable Clear.
12	at_ee_pi2_err	WRITE	AT Link 2 PI Error Enable Clear.
11	at_ee_pi1_err	WRITE	AT Link 1 PI Error Enable Clear.
10	at_ee_pi0_err	WRITE	AT Link 0 PI Error Enable Clear.
9	at_ee_rp1_sys_err	WRITE	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enable Clear.
8	at_ee_rp1_rp3_err	WRITE	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enable Clear.
7	at_ee_rp1_crc_err	WRITE	AT RP1 Type CRC Error Enable Clear.
6	at_ee_rp1_bit_width_err	WRITE	AT RP1 Type Bit Width Error Enable Clear.
5	at_ee_rp1_type_rsvd_err	WRITE	AT RP1 Type Reserved Error Enable Clear.
4	at_ee_rp1_type_spare_err	WRITE	AT RP1 Type Spare Error Enable Clear.
3	at_ee_rp1_type_unsel_err	WRITE	AT RP1 Type Unsel Error Enable Clear.
2	at_ee_rp1_type_tod_rcvd_err	WRITE	AT RP1 Type TOD Receive Error Enable Clear.
1	at_ee_rp1_type_rp3_rcvd_err	WRITE	AT RP1 Type RP3 Receive Error Enable Clear.
0	at_ee_rp1_type_sys_rcvd_err	WRITE	AT RP1 Type SYS Receive Error Enable Clear.

8.13.1.89 ee_at_en_sts_ev0 Register (offset = 0x40324)

Table 8-361. ee_at_en_sts_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	READ	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enabled.
16	at_ee_phyt_sync_err	READ	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enabled.
15	at_ee_pi5_err	READ	AT Link 5 PI Error Enabled.
14	at_ee_pi4_err	READ	AT Link 4 PI Error Enabled.
13	at_ee_pi3_err	READ	AT Link 3 PI Error Enabled.
12	at_ee_pi2_err	READ	AT Link 2 PI Error Enabled.
11	at_ee_pi1_err	READ	AT Link 1 PI Error Enabled.
10	at_ee_pi0_err	READ	AT Link 0 PI Error Enabled.
9	at_ee_rp1_sys_err	READ	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enabled.
8	at_ee_rp1_rp3_err	READ	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enabled.
7	at_ee_rp1_crc_err	READ	AT RP1 Type CRC Error Enabled.
6	at_ee_rp1_bit_width_err	READ	AT RP1 Type Bit Width Error Enabled.
5	at_ee_rp1_type_rsvd_err	READ	AT RP1 Type Reserved Error Enabled.
4	at_ee_rp1_type_spare_err	READ	AT RP1 Type Spare Error Enabled.
3	at_ee_rp1_type_unsel_err	READ	AT RP1 Type Unsel Error Enabled.
2	at_ee_rp1_type_tod_rcvd_err	READ	AT RP1 Type TOD Receive Error Enabled.
1	at_ee_rp1_type_rp3_rcvd_err	READ	AT RP1 Type RP3 Receive Error Enabled.
0	at_ee_rp1_type_sys_rcvd_err	READ	AT RP1 Type SYS Receive Error Enabled.

8.13.1.90 ee_at_en_sts_ev1 Register (offset = 0x40328)

Table 8-362. ee_at_en_sts_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-18	RSVD	NOT_ACCESSIBLE	Reserved.
17	at_ee_radt_sync_err	READ	AT RADT sync input is not aligned to the RADT counter frame boundary Error Enabled.
16	at_ee_phyt_sync_err	READ	AT PHYT sync input is not aligned to the PHYT counter frame boundary Error Enabled.
15	at_ee_pi5_err	READ	AT Link 5 PI Error Enabled.
14	at_ee_pi4_err	READ	AT Link 4 PI Error Enabled.
13	at_ee_pi3_err	READ	AT Link 3 PI Error Enabled.
12	at_ee_pi2_err	READ	AT Link 2 PI Error Enabled.
11	at_ee_pi1_err	READ	AT Link 1 PI Error Enabled.
10	at_ee_pi0_err	READ	AT Link 0 PI Error Enabled.
9	at_ee_rp1_sys_err	READ	AT RADT selected width Frame number does not match the received RP1 SYS frame number Error Enabled.
8	at_ee_rp1_rp3_err	READ	AT PHYT selected width Frame number does not match the received RP1 RP3 frame number Error Enabled.
7	at_ee_rp1_crc_err	READ	AT RP1 Type CRC Error Enabled.
6	at_ee_rp1_bit_width_err	READ	AT RP1 Type Bit Width Error Enabled.
5	at_ee_rp1_type_rsvd_err	READ	AT RP1 Type Reserved Error Enabled.
4	at_ee_rp1_type_spare_err	READ	AT RP1 Type Spare Error Enabled.
3	at_ee_rp1_type_unsel_err	READ	AT RP1 Type Unsel Error Enabled.
2	at_ee_rp1_type_tod_rcvd_err	READ	AT RP1 Type TOD Receive Error Enabled.
1	at_ee_rp1_type_rp3_rcvd_err	READ	AT RP1 Type RP3 Receive Error Enabled.
0	at_ee_rp1_type_sys_rcvd_err	READ	AT RP1 Type SYS Receive Error Enabled.

8.13.1.91 ee_pd_common_irs Register (offset = 0x40400)

Table 8-363. ee_pd_common_irs Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	READ	(**Error**) Global, Watch dog timer, time out before reception of GSM Time Slot EOP

8.13.1.92 ee_pd_common_irs_set Register (offset = 0x40404)

Table 8-364. ee_pd_common_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	WRITE	PD ts watchdog Error Set.

8.13.1.93 ee_pd_common_irs_clr Register (offset = 0x40408)

Table 8-365. ee_pd_common_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	WRITE	PD ts watchdog Error Clear.

8.13.1.94 ee_pd_common_en_ev0 Register (offset = 0x4040C)

Table 8-366. ee_pd_common_en_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	READ	PD ts watchdog Error Enable.

8.13.1.95 ee_pd_common_en_set_ev0 Register (offset = 0x40410)

Table 8-367. ee_pd_common_en_set_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	WRITE	PD ts watchdog Error Enable Set.

8.13.1.96 ee_pd_common_en_clr_ev0 Register (offset = 0x40414)

Table 8-368. ee_pd_common_en_clr_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	WRITE	PD ts watchdog Error Enable Clear.

8.13.1.97 ee_pd_common_en_ev1 Register (offset = 0x40418)

Table 8-369. ee_pd_common_en_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	READ	PD ts watchdog Error Enable.

8.13.1.98 ee_pd_common_en_set_ev1 Register (offset = 0x4041C)

Table 8-370. ee_pd_common_en_set_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	WRITE	PD ts watchdog Error Enable Set.

8.13.1.99 ee_pd_common_en_clr_ev1 Register (offset = 0x40420)

Table 8-371. ee_pd_common_en_clr_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	WRITE	PD ts watchdog Error Enable Clear.

8.13.1.100 ee_pd_common_en_sts_ev0 Register (offset = 40424)

Table 8-372. ee_pd_common_en_sts_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	READ	PD ts watchdog Error Enabled.

8.13.1.101 ee_pd_common_en_sts_ev1 Register (offset = 0x40428)

Table 8-373. ee_pd_common_en_sts_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-1	RSVD	NOT_ACCESSIBLE	Reserved.
0	pd_ee_ts_wdog_err	READ	PD ts watchdog Error Enabled.

8.13.1.102 ee_pe_common_irs_set Register (offset = 0x40500)

Table 8-374. ee_pe_common_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	READ	(**Error**), Data request FIFO overflow
2	pe_ee_token_wr_err	READ	(**Information**), One or more DMA transfer tokens has been issued by PE to DB (for debug)
1	pe_ee_token_req_ovfl_err	READ	(**Error**), Token request FIFO overflow.
0	pe_ee_rd2db_err	READ	(**Information)**, If any DB channel is read and DB returns data (ACK). Indicating PE is both processing traffic and DB is supplying data (for debug)

8.13.1.103 ee_pe_common_irs_set Register (offset = 0x40504)

Table 8-375. ee_pe_common_irs_set Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	WRITE	PE Data Request Overflow Error Set.
2	pe_ee_token_wr_err	WRITE	PE Token Write Error Set
1	pe_ee_token_req_ovfl_err	WRITE	PE Token Request Overflow Error Set.
0	pe_ee_rd2db_err	WRITE	PE Read to DB Error Set.

8.13.1.104 ee_pe_common_irs_clr Register (offset = 0x40508)

Table 8-376. ee_pe_common_irs_clr Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	WRITE	PE Data Request Overflow Error Clear.
2	pe_ee_token_wr_err	WRITE	PE Token Write Error Clear
1	pe_ee_token_req_ovfl_err	WRITE	PE Token Request Overflow Error Clear.
0	pe_ee_rd2db_err	WRITE	PE Read to DB Error Clear.

8.13.1.105 ee_pe_common_en_ev0 Register (offset = 0x4050C)

Table 8-377. ee_pe_common_en_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	READ	PE Data Request Overflow Error Enable.
2	pe_ee_token_wr_err	READ	PE Token Write Error Enable.
1	pe_ee_token_req_ovfl_err	READ	PE Token Request Overflow Error Enable.
0	pe_ee_rd2db_err	READ	PE Read to DB Error Enable.

8.13.1.106 ee_pe_common_en_set_ev0 Register (offset = 0x40510)

Table 8-378. ee_pe_common_en_set_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	WRITE	PE Data Request Overflow Error Set.
2	pe_ee_token_wr_err	WRITE	PE Token Write Error Set
1	pe_ee_token_req_ovfl_err	WRITE	PE Token Request Overflow Error Set.
0	pe_ee_rd2db_err	WRITE	PE Read to DB Error Set.

8.13.1.107 ee_pe_common_en_clr_ev0 Register (offset = 0x40514)

Table 8-379. ee_pe_common_en_clr_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	WRITE	PE Data Request Overflow Error Clear.
2	pe_ee_token_wr_err	WRITE	PE Token Write Error Clear
1	pe_ee_token_req_ovfl_err	WRITE	PE Token Request Overflow Error Clear.
0	pe_ee_rd2db_err	WRITE	PE Read to DB Error Clear.

8.13.1.108 ee_pe_common_en_ev1 Register (offset = 0x40518)

Table 8-380. ee_pe_common_en_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	READ	PE Data Request Overflow Error Enable.
2	pe_ee_token_wr_err	READ	PE Token Write Error Enable.
1	pe_ee_token_req_ovfl_err	READ	PE Token Request Overflow Error Enable.
0	pe_ee_rd2db_err	READ	PE Read to DB Error Enable.

8.13.1.109 ee_pe_common_en_set_ev1 Register (offset = 0x4051C)

Table 8-381. ee_pe_common_en_set_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	WRITE	PE Data Request Overflow Error Set.
2	pe_ee_token_wr_err	WRITE	PE Token Write Error Set
1	pe_ee_token_req_ovfl_err	WRITE	PE Token Request Overflow Error Set.
0	pe_ee_rd2db_err	WRITE	PE Read to DB Error Set.

8.13.1.110 ee_pe_common_en_clr_ev1 Register (offset = 0x40520)

Table 8-382. ee_pe_common_en_clr_ev1 Register

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	WRITE	PE Data Request Overflow Error Clear.
2	pe_ee_token_wr_err	WRITE	PE Token Write Error Clear
1	pe_ee_token_req_ovfl_err	WRITE	PE Token Request Overflow Error Clear.
0	pe_ee_rd2db_err	WRITE	PE Read to DB Error Clear.

8.13.1.111 ee_pe_common_en_sts_ev0 Register (offset = 0x40254)

Table 8-383. ee_pe_common_en_sts_ev0 Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	READ	PE Data Request Overflow Error Enabled.
2	pe_ee_token_wr_err	READ	PE Token Write Error Enabled.
1	pe_ee_token_req_ovfl_err	READ	PE Token Request Overflow Error Enabled.
0	pe_ee_rd2db_err	READ	PE Read to DB Error Enabled.

8.13.1.112 ee_pe_common_en_sts_ev1 Register (offset = 0x40528)

Table 8-384. ee_pe_common_en_sts_ev1 Register Field Descriptions

Bits	Field Name	Type	Description
31-4	RSVD	NOT_ACCESSIBLE	Reserved.
3	pe_ee_dat_req_ovfl_err	READ	PE Data Request Overflow Error Enabled.
2	pe_ee_token_wr_err	READ	PE Token Write Error Enabled.
1	pe_ee_token_req_ovfl_err	READ	PE Token Request Overflow Error Enabled.
0	pe_ee_rd2db_err	READ	PE Read to DB Error Enabled.

AIF2 Example Configuration

This chapter shows some useful example configurations about Transmission rules, Data buffer setup, AIF2 DMA and PKTDMA setup, and AIF2 timer setup for WCDMA, LTE cases and Generic packet traffic. This example shows only how to configure PD, PE, RT, DB, AD, AT, and PKTDMA modules; it does not include SD, RM, TM, CI, or CO configurations. Users can get details about these module setups from the AIF2 CSL example code.

Topic **Page**

9.1 Transmission Rule Setup (PE Setup) 623
9.2 Up Link Ingress Setup (PD, RT Setup) 629
9.3 Direct IO Setup 631
9.4 AIF2 PKTDMA and QM Setup 633
9.5 Generic Packet Traffic Setup 638

9.1 Transmission Rule Setup (PE Setup)

9.1.1 WCDMA OBSAI 4x, DL, 16 AxC With Four Control Channels (Four DSPs in Chain Topology)

```
//PE link setup for all DSPs in the chain
PeLinkSetup.PeCppiDioSel = CSL_AIF2_DIO;
PeLinkSetup.PeDelay = 28; // sys_clks delay between DB and PE

//PE common setup for all DSPs in the chain
PeCommonSetup.PeTokenPhase = 0;
PeCommonSetup.GlobalDioLen = CSL_AIF2_DB_DIO_LEN_128;
PeCommonSetup.PeFrameTC[0].FrameIndexSc = 0; //start index
PeCommonSetup.PeFrameTC[0].FrameIndexTc = 0; //terminal index
PeCommonSetup.PeFrameTC[0].FrameSymbolTc = 14;//Symbol index
PeCommonSetup.PeFrameMsgTc[0] = 639; // WCDMA has 640 OBSAI messages in one slot time

for(i=0;i<16;i++)//for AxC channel 0 ~ 15
{
PeCommonSetup.bEnableCh[i] = TRUE;//Enable PE channel
PeCommonSetup.PeDmaCh0[i].bCrcEn = FALSE;//disable CRC
PeCommonSetup.PeDmaCh0[i].FrameTC = 0;//use framing terminal count 0
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_INSERT;//for first DL DSP in the chain
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_ADD16;//for all other DL DSP in the chain
PeCommonSetup.PeDmaCh0[i].isEthernet = FALSE;//AxC data
PeCommonSetup.PeInFifo[i].SyncSymbol = 0;//Sync symbol offset
PeCommonSetup.PeInFifo[i].MFifoWmark = 2;//Message FIFO water mark
PeCommonSetup.PeInFifo[i].MFifoFullLevel = 3;//Message FIFO full level
PeCommonSetup.PeAxcOffset[i] = 610;//PE2 event offset + external AxC offset ==> 610 + 0
PeCommonSetup.PeChObsaiType[i] = OBSAI_TYPE_WCDMA_FDD;
PeCommonSetup.PeChObsaiTS[i] = 0;//automatically inserted by HW
PeCommonSetup.PeChObsaiAddr[i] = i;//OBSAI header address for 16 channels
PeCommonSetup.PeChObsaiTsMask[i] = CSL_AIF2_ROUTE_MASK_NONE;//not use PeChObsaiTS field
PeCommonSetup.PeChObsaiTsfomat[i] = CSL_AIF2_TSTAMP_FORMAT_NORM_TS;//OBSAI header TS format
PeCommonSetup.PeObsaiPkt[i] = FALSE;//OBSAI channel is used for AxC data
}

//Only one modulo rule assigned for 16 AxC channels
PeCommonSetup.PeModuloTc[0].bEnableRule = TRUE;
PeCommonSetup.PeModuloTc[0].RuleModulo = 0;
PeCommonSetup.PeModuloTc[0].bRuleObsaiCtlMsg = FALSE;//OBSAI AxC data
PeCommonSetup.PeModuloTc[0].RuleIndex = 0;
PeCommonSetup.PeModuloTc[0].RuleLink = CSL_AIF2_LINK_0;
//DBM rule 0 is working on Modulo rule 0 and control 16 AxC data
PeCommonSetup.PeObsaiDualBitMap[0].DbmX = 15;//set X-1 value
PeCommonSetup.PeObsaiDualBitMap[0].DbmXBubble = 0;
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Mult = 0;
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Size = 0;
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Map[0] = 0x0;// no bubble
PeCommonSetup.PeObsaiDualBitMap[0].Dbm2Size = 0;
PeCommonSetup.PeObsaiDualBitMap[0].Dbm2Map[0] = 0x0;


for (i=0; i<16; i++)//Channel index LUT setup for 16 AxC channels
{
PeCommonSetup.ChIndex0[i] = i; //match each DBM X index to channel 0 ~ 15
PeCommonSetup.bEnableChIndex0[i] = TRUE;
}

For Control message, each DSP in the chain can use different channel or same channel with
different OBSAI address. This example shows the different channel case. so user can setup channel
16 for DSP0, 17 for DSP1, 18 for DSP2, 19 for DSP3 in the chain

Control message channel common setup is basically same to AxC channels except
PeCommonSetup.PeDmaCh0[16 ~ 19].RtControl = CSL_AIF2_PE_RT_INSERT; and
PeCommonSetup.PeObsaiPkt[16 ~ 19] = TRUE; for all DSPs in the chain
```

```
//Use modulo rule 1 for 4 Control channels
PeCommonSetup.PeModuloTc[1].bEnableRule = TRUE;
PeCommonSetup.PeModuloTc[1].RuleModulo = 0;
PeCommonSetup.PeModuloTc[1].bRuleObsaiCtlMsg = TRUE;//OBSAI Control msg
PeCommonSetup.PeModuloTc[1].RuleIndex = 0;
PeCommonSetup.PeModuloTc[1].RuleLink = CSL_AIF2_LINK_0;
//DBM rule 1is working on Modulo rule 1and DbmX is 4 this time
PeCommonSetup.PeObsaiDualBitMap[1].DbmX = 3;//set X-1 value
PeCommonSetup.PeObsaiDualBitMap[1].DbmXBubble = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm1Mult = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm1Size = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm1Map[0] = 0x0;// no bubble
PeCommonSetup.PeObsaiDualBitMap[1].Dbm2Size = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm2Map[0] = 0x0;

Channel index LUT setup is different for each DSP in the chain. DSP0 uses channel 16 while DSP1
used channel 17......
PeCommonSetup.ChIndex0[64] = 16;
PeCommonSetup.bEnableChIndex0[64] = TRUE; // only for DSP0
PeCommonSetup.ChIndex0[65] = 17;
PeCommonSetup.bEnableChIndex0[65] = TRUE; // only for DSP1
PeCommonSetup.ChIndex0[66] = 18;
PeCommonSetup.bEnableChIndex0[66] = TRUE; // only for DSP2
PeCommonSetup.ChIndex0[67] = 19;
PeCommonSetup.bEnableChIndex0[67] = TRUE; // only for DSP3
```

Each DSP requires Multicore Navigator configuration for each control channel data DMA. See Section 9.4 to get more details.

9.1.2 WCDMA CPRI 4x, DL, 16 AxC With One Control Channel (Four DSPs in Chain Topology)

CPRI and OBSAI have many differences when configuring transmission rules. First, CPRI does not have module rules and only one DBMR can be assigned for each link for AxC data or Packet data.

The *Control Words* bandwidth is controlled by a CPRI CW 256 LUT. Within each *basic frame*, you would allocate which control words to be used for which DSP in the chain and furthermore, control which of 4 “packing channels” would fill that control word.

PE Channel Rule LUT register has cpri_pkt_en field. If it is set to zero, the channel will be used only for AxC data which has symbol or chip boundary. If it is set to one, that channel will be used to transfer Generic, Ethernet or CPRI control word. Each link uses maximum 128 channel rule LUT.

```
//PE link setup
PeLinkSetup.PeCppiDioSel = CSL_AIF2_DIO;
PeLinkSetup.PeDelay = 0;//sys_clks delay between DB and PE
PeLinkSetup.PeCpriDualBitMap.DbmX = 15;//set X-1
PeLinkSetup.PeCpriDualBitMap.DbmXBubble = 1;//2 bubbles of 1 AxC sample
PeLinkSetup.PeCpriDualBitMap.Dbm1Mult = 0;//set n-1
PeLinkSetup.PeCpriDualBitMap.Dbm1Size = 0;//set n-1
PeLinkSetup.PeCpriDualBitMap.Dbm1Map[0] = 0x0;
PeLinkSetup.PeCpriDualBitMap.Dbm2Size = 0;
PeLinkSetup.PeCpriDualBitMap.Dbm2Map[0] = 0x0;
PeLinkSetup.CpriAxCPack = CSL_AIF2_CPRI_15BIT_SAMPLE;
PeLinkSetup.CpriCwNullDelimitor = 0xFB;//K 27.7 character
PeLinkSetup.CpriCwPktDelimitor[0] = CSL_AIF2_CW_DELIM_NULLDELM;
PeLinkSetup.PePackDmaCh[0] = 124; // use channel 124 for control word
PeLinkSetup.bEnablePack[0] = TRUE;
for(i=0;i<256;i++)//cpri cw lut setup
{
PeLinkSetup.CpriCwChannel[i]= 0; /* set cw packing channel num to 0. max four packing
             channels could be assigned for each DSP to transfer control word */
PeLinkSetup.bEnableCpriCw[i]= TRUE;
}
//PE common setup
PeCommonSetup.PeTokenPhase = 0;
PeCommonSetup.EnetHeaderSelect = 0;//bit order for Ethernet preamble and SOF
PeCommonSetup.GlobalDioLen = CSL_AIF2_DB_DIO_LEN_128;
PeCommonSetup.PeFrameTC[0].FrameSymbolTc = 14;//Symbol index
for(i=0;i<16;i++)//for channel 0 ~ 15
{
PeCommonSetup.bEnableCh[i] = TRUE;//Enable AxC channel
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_INSERT;//for first DL DSP in the chain
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_ADD15;//for all other DL DSP in the chai
PeCommonSetup.PeDmaCh0[i].CrcType = CSL_AIF2_CRC_8BIT;//CRC type
PeCommonSetup.PeDmaCh0[i].isEthernet = FALSE;//AxC data
PeCommonSetup.PeInFifo[i].SyncSymbol = 0;//Sync symbol offset
PeCommonSetup.PeInFifo[i].MFifoWmark = 2;//Message FIFO water mark
PeCommonSetup.PeInFifo[i].MFifoFullLevel = 3;//Message FIFO full level
PeCommonSetup.PeAxcOffset[i] = 310;//PE2 offset + No external AxC offset
PeCommonSetup.PeFrameMsgTc[0] = 2559;// 2560 CPRI samples (4 byte) are in WCDMA slot time
//PE Channel LUT setup and link routing selection
PeCommonSetup.ChIndex1[0 ~ 15] = 0 ~ 15; //channel 0 ~ channel 15 for AxC data
PeCommonSetup.bEnableChIndex1[0 ~ 15] = TRUE;//Route egress dbm rule to link1
PeCommonSetup.CpriPktEn1[0 ~ 15] = FALSE;//AxC data
PeCommonSetup.ChIndex1[124] = 124; //channel 124 for control word
PeCommonSetup.bEnableChIndex1[124] = TRUE;//Route egress channel rule to link1
PeCommonSetup.CpriPktEn1[124] = TRUE;//Control data for each DSP
```

9.1.3 20 MHz LTE OBSAI 4x,DL, 2 AxC With Two Control Channels (Two DSPs in Chain Topology)

```
//PE link setup
PeLinkSetup.PeCppiDioSel = CSL_AIF2_CPPI;
PeLinkSetup.TddAxc = FALSE;
PeLinkSetup.bEnObsaiBubbleBW = FALSE;
PeLinkSetup.PeDelay = 28;//sys_clks delay between DB and PE
//PE common setup
PeCommonSetup.PeTokenPhase = 0;
PeCommonSetup.PeFrameTC[0].FrameIndexSc = 0;//start index
PeCommonSetup.PeFrameTC[0].FrameIndexTc = 5;//terminal index
PeCommonSetup.PeFrameTC[0].FrameSymbolTc = 119;// 120 extended cyclic prefix symbols
            in one rad frame.
for(i=0;i<2;i++)//for AxC channel 0,1
{
PeCommonSetup.bEnableCh[i] = 0;//Enable PE channel
PeCommonSetup.PeDmaCh0[i].bCrcEn = FALSE;//disable CRC
PeCommonSetup.PeDmaCh0[i].FrameTC = 0;//use framing terminal count 0
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_INSERT;//for first DL DSP in the chain
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_ADD16;//for second DSP in the chain
PeCommonSetup.PeDmaCh0[i].isEthernet = FALSE;//AxC data
PeCommonSetup.PeDmaCh0[i].CrcObsaiHeader = FALSE;
PeCommonSetup.PeInFifo[i].SyncSymbol = 0;//sync symbol offset
PeCommonSetup.PeInFifo[i].MFifoWmark = 3;//Message FIFO water mark
PeCommonSetup.PeInFifo[i].MFifoFullLevel = 5;//Message FIFO full level
PeCommonSetup.PeAxcOffset[i] = 310; //PE2 event offset + External offset = 310 + 0
}

for(i=0;i<6;i++)
PeCommonSetup.PeFrameMsgTc[i] = 639;// One LTE cyclic extended prefix symbol has 640 OBSAI
messages
PeCommonSetup.PeModuloTc[0].bEnableRule = TRUE;
PeCommonSetup.PeModuloTc[0].RuleModulo = 0;//Set modulo minus one
PeCommonSetup.PeModuloTc[0].bRuleObsaiCtlMsg = FALSE;
PeCommonSetup.PeModuloTc[0].RuleIndex = 0;//always fire
PeCommonSetup.PeModuloTc[0].RuleLink = CSL_AIF2_LINK_0;

for(i=0; i < 2; i++){
PeCommonSetup.PeChObsaiType[i] = OBSAI_TYPE_LTE;//OBSAI header type
PeCommonSetup.PeChObsaiTS[i] = 0;//OBSAI header Timestamp
PeCommonSetup.PeChObsaiAddr[i] = i;//OBSAI header address
PeCommonSetup.PeChObsaiTsMask[i] = CSL_AIF2_ROUTE_MASK_NONE;
PeCommonSetup.PeChObsaiTsfomat[i] = CSL_AIF2_TSTAMP_FORMAT_NORM_TS;
PeCommonSetup.PeObsaiPkt[i] = FALSE;//Select OBSAI AxC mode
}

//Dual bit map setup for AxC channels
PeCommonSetup.PeObsaiDualBitMap[0].DbmX = 1;//set X-1
PeCommonSetup.PeObsaiDualBitMap[0].DbmXBubble = 0;
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Mult = 0;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Size = 0;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Map[0] = 0x0;
PeCommonSetup.ChIndex0[0] = 0; //AxC channel 0 for DSP0 in the chain
PeCommonSetup.bEnableChIndex0[0] = TRUE;/* Route egress channel 0 to modulo rule 0. only DSP0
should have this setup */
PeCommonSetup.ChIndex0[1] = 1; //AxC channel 1for DSP1 in the chain
PeCommonSetup.bEnableChIndex0[1] = TRUE;/* Route egress channel 1to modulo rule 0.
            only DSP1 should have this setup */
For Control message, each DSP in the chain can use different channel or same channel with
different OBSAI address. this example shows the different channel case. so user can setup
            channel 2 for DSP0, channel 3 for DSP1.
Control message channel common setup is basically same to AxC channels except
PeCommonSetup.PeDmaCh0[2 ~ 3].RtControl = CSL_AIF2_PE_RT_INSERT; and PeCommonSetup.PeObsaiPkt[2 ~
            3] = TRUE; for all DSPs in the chain.
//Use modulo rule 1 for 2Control channels
PeCommonSetup.PeModuloTc[1].bEnableRule = TRUE;
```

```
PeCommonSetup.PeModuloTc[1].RuleModulo = 0;
PeCommonSetup.PeModuloTc[1].bRuleObsaiCtlMsg = TRUE;//OBSAI Control msg
PeCommonSetup.PeModuloTc[1].RuleIndex = 0;
PeCommonSetup.PeModuloTc[1].RuleLink = CSL_AIF2_LINK_0;
//DBM rule 1is working on Modulo rule 1and X is 2 this time
PeCommonSetup.PeObsaiDualBitMap[1].DbmX = 1;//set X-1 value
PeCommonSetup.PeObsaiDualBitMap[1].DbmXBubble = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm1Mult = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm1Size = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm1Map[0] = 0x0;// no bubble
PeCommonSetup.PeObsaiDualBitMap[1].Dbm2Size = 0;
PeCommonSetup.PeObsaiDualBitMap[1].Dbm2Map[0] = 0x0;
Channel index LUT setup is different for each DSP in the chain. DSP0 uses channel 2 while DSP1
used channel 3.
PeCommonSetup.ChIndex0[64] = 2;
PeCommonSetup.bEnableChIndex0[64] = TRUE; // only for DSP0
PeCommonSetup.ChIndex0[65] = 3;
PeCommonSetup.bEnableChIndex0[65] = TRUE; // only for DSP1
```

9.1.4 20 MHz LTE CPRI 4x, DL, 2 AxC With One Control Channel (Two DSPs in Chain Topology)

```
//PE link setup for channel 0 and 1
PeLinkSetup.PeCppiDioSel = CSL_AIF2_CPPI;
PeLinkSetup.PeDelay = 0;// sys_clks delay between DB and PE
PeLinkSetup.PeCpriDualBitMap.DbmX = 1;//assign sample place for 2 AxC channel. Set X-1
PeLinkSetup.PeCpriDualBitMap.DbmXBubble = 1;//2 bubbles of 1 AxC sample
PeLinkSetup.PeCpriDualBitMap.Dbm1Mult = 0;//set n-1
PeLinkSetup.PeCpriDualBitMap.Dbm1Size = 0;//set n-1
PeLinkSetup.PeCpriDualBitMap.Dbm1Map[0] = 0x0;
PeLinkSetup.PeCpriDualBitMap.Dbm2Size = 0;
PeLinkSetup.PeCpriDualBitMap.Dbm2Map[0] = 0x0;
PeLinkSetup.CpriAxCPack = CSL_AIF2_CPRI_15BIT_SAMPLE;
PeLinkSetup.CpriCwNullDelimitor = 0xFB;// K 27.7 character
PeLinkSetup.CpriCwPktDelimitor[0] = CSL_AIF2_CW_DELIM_NULLDELM;
PeLinkSetup.PePackDmaCh[0] = 124;
PeLinkSetup.bEnablePack[0] = TRUE;
for(i=0;i<256;i++)/* cpri cw lut setup. it could be differently setup for DSP0 and DSP1 to split
control word BW */
{
PeLinkSetup.CpriCwChannel[i]= 0; /* set cw packing channel num to 0. four packing channels could
be assigned for each DSP to transfer control word */
PeLinkSetup.bEnableCpriCw[i]= TRUE;
}

//PE common setup
PeCommonSetup.PeTokenPhase = 0;
PeCommonSetup.EnetHeaderSelect = 0;//bit order for Ethernet preamble and SOF
PeCommonSetup.PeFrameTC[0].FrameIndexSc = 0;//start index
PeCommonSetup.PeFrameTC[0].FrameIndexTc = 5;//terminal index
PeCommonSetup.PeFrameTC[0].FrameSymbolTc = 119;// 120 cyclic extended prefix symbols

for(i=0;i<2;i++)//for channel 0 and 1
{
PeCommonSetup.bEnableCh[i] = TRUE;//Enable AxC channel
PeCommonSetup.PeDmaCh0[i].FrameTC = 0;
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_INSERT;//for first DL DSP in the chain
PeCommonSetup.PeDmaCh0[i].RtControl = CSL_AIF2_PE_RT_ADD16;//for second DSP in the chain
PeCommonSetup.PeDmaCh0[i].isEthernet = FALSE;//AxC data
PeCommonSetup.PeInFifo[i].SyncSymbol = 0;//Sync symbol offset
PeCommonSetup.PeInFifo[i].MFifoWmark = 3;//Message FIFO water mark
PeCommonSetup.PeInFifo[i].MFifoFullLevel = 5;//Message FIFO full level
PeCommonSetup.PeAxcOffset[i] = 310; //PE2 offset + No external line delay
}

for(i=0;i<6;i++)
```

```
PeCommonSetup.PeFrameMsgTc[i] = 2559;// One LTE extended cyclic prefix symbol has 2560 samples (4
byte) //PE Channel LUT setup and link routing selection
PeCommonSetup.ChIndex1[0] = 0; //channel 0 only for DSP0
PeCommonSetup.bEnableChIndex1[0] = TRUE;//Route egress dbm rule to link1
PeCommonSetup.CpriPktEn1[0] = FALSE;//AxC data
PeCommonSetup.ChIndex1[1] = 1; //channel 1only for DSP1
PeCommonSetup.bEnableChIndex1[1] = TRUE;//Route egress dbm rule to link1
PeCommonSetup.CpriPktEn1[1] = FALSE;//AxC data
PeCommonSetup.ChIndex1[124] = 124; //channel 124 for both DSP
PeCommonSetup.bEnableChIndex1[124] = TRUE;//Route egress channel rule to link1
PeCommonSetup.CpriPktEn1[124] = TRUE;//Control data for each DSP
```

9.1.5 15 MHz LTE OBSAI With Two Different Options

Option1: Insert bubbles by DBMR rule setup. (8x link speed with 5 AxCs)

```
//PD frame message TC setup
PdCommonSetup.PdFrameMsgTc[0] = 413;
for(i=1;i<7;i++)
PdCommonSetup.PdFrameMsgTc[i] = 410;//terminal count is smaller than 20 MHz case
//PE frame message TC setup
PeCommonSetup.PeFrameMsgTc[0] = 413;
for(i=1;i<7;i++)
PeCommonSetup.PeFrameMsgTc[i] = 410;
//Dual bit map setup for LTE 15 MHz. map size is three and insert bubble every second iteration.
PeCommonSetup.PeObsaiDualBitMap[0].DbmX = 4;//set X-1.
            PeCommonSetup.PeObsaiDualBitMap[0].DbmXBubble = 0;//OBSAI 4 AxC sample length bubble
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Mult = 0;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Size = 2;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Map[0] = 0x2;
PeCommonSetup.ChIndex0[0] = 0; //channel 0
PeCommonSetup.bEnableChIndex0[0] = TRUE;
PeCommonSetup.ChIndex0[1] = 1; //channel 1
PeCommonSetup.bEnableChIndex0[1] = TRUE;
PeCommonSetup.ChIndex0[2] = 2; //channel 2
PeCommonSetup.bEnableChIndex0[2] = TRUE;
PeCommonSetup.ChIndex0[3] = 3; //channel 3
PeCommonSetup.bEnableChIndex0[3] = TRUE;
PeCommonSetup.ChIndex0[4] = 4; //channel 4
PeCommonSetup.bEnableChIndex0[4] = TRUE;
Option2: Shut down PE Channel LUT for unused 25% BW. (4x link speed with 2 AxCs)
//PD frame message TC setup
PdCommonSetup.PdFrameMsgTc[0] = 413;

for(i=1;i<7;i++)
PdCommonSetup.PdFrameMsgTc[i] = 410;//terminal count is smaller than 20 MHz case
//PE frame message TC setup
PeCommonSetup.PeFrameMsgTc[0] = 413;
for(i=1;i<7;i++)
PeCommonSetup.PeFrameMsgTc[i] = 410
//Dual bit map setup for LTE 15 MHz. No bubble option
PeCommonSetup.PeObsaiDualBitMap[0].DbmX = 7;//set X-1. Eight message slot for only 2 AxCs
PeCommonSetup.PeObsaiDualBitMap[0].DbmXBubble = 0;
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Mult = 0;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Size = 0;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Map[0] = 0x0;
//PE Channel LUT setup
//{AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1}
PeCommonSetup.ChIndex0[0] = 0;
PeCommonSetup.bEnableChIndex0[0] = TRUE;
PeCommonSetup.ChIndex0[1] = 1;
PeCommonSetup.bEnableChIndex0[1] = TRUE;
PeCommonSetup.ChIndex0[2] = 0;
PeCommonSetup.bEnableChIndex0[2] = TRUE;
PeCommonSetup.ChIndex0[3] = 1;
PeCommonSetup.bEnableChIndex0[3] = TRUE;
```

```
PeCommonSetup.ChIndex0[4] = 0;
PeCommonSetup.bEnableChIndex0[4] = TRUE;
PeCommonSetup.ChIndex0[5] = 1;
PeCommonSetup.bEnableChIndex0[5] = TRUE;
PeCommonSetup.ChIndex0[6] = 0;
PeCommonSetup.bEnableChIndex0[6] = FALSE;// turn off 7th and 8th message slot
PeCommonSetup.ChIndex0[7] = 1;
PeCommonSetup.bEnableChIndex0[7] = FALSE;
```

In this case, shut down message slot will send empty OBSAI message (all header is 0xFFFFFF and payload is zero)

9.2 Up Link Ingress Setup (PD, RT Setup)

9.2.1 WCDMA OBSAI 2x, 8 AxC With Retransmission

```
//PD link setup
PdLinkSetup.PdTypeLut[OBSAI_TYPE_WCDMA_FDD].ObsaiTsFormat = CSL_AIF2_TSTAMP_FORMAT_NORM_TS;
PdLinkSetup.PdTypeLut[OBSAI_TYPE_WCDMA_FDD].PdCrcType = CSL_AIF2_CRC_8BIT;
PdLinkSetup.PdTypeLut[OBSAI_TYPE_WCDMA_FDD].bEnableCrc = TRUE;
PdLinkSetup.PdTypeLut[OBSAI_TYPE_WCDMA_FDD].PdObsaiMode = CSL_AIF2_PD_DATA_AXC;
PdLinkSetup.PdTypeLut[OBSAI_TYPE_WCDMA_FDD].bEnableEnetStrip = FALSE;
PdLinkSetup.PdTypeLut[OBSAI_TYPE_WCDMA_FDD].bEnableCrcHeader = FALSE; //OBSAI crc calculation for
header
//PD common setup
PdCommonSetup.PdCppiDioSel = CSL_AIF2_DIO;//DIO
PdCommonSetup.AxCOffsetWin = 300; // +/-
 300 byte clock size window for first valid AxC data detection
PdCommonSetup.PdRadtTC = 3071999; // Radio frame size for OBSAI
PdCommonSetup.PdFrameTC[0].FrameIndexSc = 0;//start index
PdCommonSetup.PdFrameTC[0].FrameIndexTc = 0;//terminal index
PdCommonSetup.PdFrameTC[0].FrameSymbolTc = 14;//Symbol index
for(i=0;i<8;i++)//for channel 0 ~ 7
{
PdCommonSetup.PdRoute[i].RouteTs = 0;//Route OBSAI timestamp
PdCommonSetup.PdRoute[i].RouteType = OBSAI_TYPE_WCDMA_FDD;
PdCommonSetup.PdRoute[i].RouteAddr = i;//Route OBSAI address
PdCommonSetup.PdRoute[i].RouteLink = CSL_AIF2_LINK_1;//Route link
PdCommonSetup.PdRoute[i].RouteMask = CSL_AIF2_ROUTE_MASK_NONE;//Route TS mask
PdCommonSetup.PdChConfig[i].bChannelEn = TRUE;//Channel enable
PdCommonSetup.PdChConfig[i].DataFormat = CSL_AIF2_LINK_DATA_TYPE_RSA;//UL Data format
PdCommonSetup.AxCOffset[i] = 0;//first valid data slot time delay from Radio boundary.
PdCommonSetup.PdChConfig1[i].bTsWatchDogEn = FALSE;//disable watchdog
PdCommonSetup.PdChConfig1[i].FrameCounter = 0;
PdCommonSetup.PdChConfig1[i].DioOffset = 0;// 8 chip size offset for UL
PdCommonSetup.PdChConfig1[i].TddEnable = 0xFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable1[i] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable2[i] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable3[i] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable4[i] = 0xFFFFFFFF;//enables all symbols(FDD)
}

PdCommonSetup.PdFrameMsgTc[0] = 639; //One WCDMA slot has 640 OBSAI messages per AxC
//RT link setup for retransmission
RtLinkSetup.CiSelect = CSL_AIF2_LINK_1;//retransmit ingress link1 data to egress
RtLinkSetup.bEnableEmptyMsg = TRUE;
RtLinkSetup.RtConfig = CSL_AIF2_RT_MODE_RETRANSMIT;// retransmission. Data goes to PD and RT at
the same time
```

9.2.2 WCDMA CPRI 2x, 8 AxC With Aggregation

```
//PD link setup
PdLinkSetup.CpriEnetStrip = 1;//enable ethernet strip for control channel 0
PdLinkSetup.Crc8Poly = CRC8_POLY;
PdLinkSetup.Crc8Seed = CRC8_SEED;
PdLinkSetup.CpriCwNullDelimitor = 0xFB; //K 27.7 character
PdLinkSetup.CpriCwPktDelimitor[0] = CSL_AIF2_CW_DELIM_NULLDELM;
PdLinkSetup.PdCpriCrcType[0] = CSL_AIF2_CRC_8BIT;
PdLinkSetup.bEnableCpriCrc[0] = TRUE;//enable CPRI CRC for control channel 0
PdLinkSetup.PdPackDmaCh[0] = 124;//Set DB channel 124 as a dma ch for control channel 0
PdLinkSetup.bEnablePack[0] = TRUE;//enable CPRI control channel 0 packing
PdLinkSetup.PdCpriDualBitMap.DbmX = 7;//set X-1
PdLinkSetup.PdCpriDualBitMap.DbmXBubble = 1;//2 bubbles of 1 AxC sample
PdLinkSetup.PdCpriDualBitMap.Dbm1Mult = 0;//set n-1
PdLinkSetup.PdCpriDualBitMap.Dbm1Size = 0;//set n-1
PdLinkSetup.PdCpriDualBitMap.Dbm1Map[0] = 0x0;
PdLinkSetup.PdCpriDualBitMap.Dbm2Size = 0;
PdLinkSetup.PdCpriDualBitMap.Dbm2Map[0] = 0x0;
for(i=0;i<8;i++)//cpri id lut setup for channel 0 ~ 7
{
PdLinkSetup.CpriDmaCh[i]= i; //DbmX channel num. this case DBMX num is matched with DB channel
num
PdLinkSetup.bEnableCpriX[i]= TRUE; //enable CPRI X
PdLinkSetup.bEnableCpriPkt[i]= FALSE;//use AxC data mode
PdLinkSetup.Cpri8WordOffset[i]= 0;//fine CPRI AxC offset
}

for(i=0;i<256;i++)//cpri cw lut setup
{
PdLinkSetup.CpriCwChannel[i]= 0; //set cw packing channel num to 0
PdLinkSetup.bEnableCpriCw[i]= TRUE; //enable CPRI X
}

//PD common setup
PdCommonSetup.PdCppiDioSel = CSL_AIF2_DIO;//DIO
PdCommonSetup.PdRadtTC = 2457599;// Radio frame size for CPRI
PdCommonSetup.PdFrameTC[0].FrameSymbolTc = 14;//Symbol index

for(i=0;i<8;i++)//for channel 0 ~ 7
{
PdCommonSetup.PdChConfig[i].bChannelEn = TRUE;//Channel enable
PdCommonSetup.PdChConfig[i].DataFormat = CSL_AIF2_LINK_DATA_TYPE_RSA;//UL Data format
PdCommonSetup.AxCOffset[i] = 0;//Not used for WCDMA
PdCommonSetup.PdChConfig1[i].bTsWatchDogEn = FALSE;//disable watchdog
PdCommonSetup.PdChConfig1[i].FrameCounter = 0;
PdCommonSetup.PdChConfig1[i].DioOffset = 0;// 8 chip size offset for UL
PdCommonSetup.PdChConfig1[i].TddEnable = 0xFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable1[i] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable2[i] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable3[i] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable4[i] = 0xFFFFFFFF;//enables all symbols(FDD)
}

PdCommonSetup.PdFrameMsgTc[0] = 639; //One WCDMA slot has 640 CPRI quad samples(16 bytes) per AxC
//RT link setup for aggregation
RtLinkSetup.CiSelect = CSL_AIF2_LINK_1;//retransmit ingress data link
RtLinkSetup.bEnableEmptyMsg = TRUE;
RtLinkSetup.RtConfig = CSL_AIF2_RT_MODE_AGGREGATE;//aggregation
```

9.3 Direct IO Setup

9.3.1 WCDMA Two Ingress DIO Engine Setup For RAC A, B Broadcast For Three AxC Channels

```
//AD Common and DIO setup
AdCommonSetup.IngrGlobalEnable = TRUE;
AdCommonSetup.IngrGlobalDioEnable = TRUE;
AdCommonSetup.FailMode = CSL_AIF2_AD_DROP;//drop fail packet
AdCommonSetup.IngrPriority = CSL_AIF2_AD_DIO_PRI;

//Ingress DIO Engine 0 setup for RAC_A
AdDioSetup.IngrDioEngineEnable[0] = TRUE;//Enable DIO Engine 0
AdDioSetup.IngrDioEngine[0].BcnTableSelect = CSL_AIF2_AD_DIO_BCN_TABLE_0;
AdDioSetup.IngrDioEngine[0].NumQuadWord = CSL_AIF2_AD_2QUAD;
AdDioSetup.IngrDioEngine[0].NumAxC = 2;//Use 3 AxC.Set N-1
AdDioSetup.IngrDioEngine[0].bEnDmaChannel = TRUE; //Enable Dma channel
AdDioSetup.IngrDioEngine[0].DmaNumBlock = 3;//4 block wrap value (RAC loop) set N-1
AdDioSetup.IngrDioEngine[0].DmaBurstLen = CSL_AIF2_AD_4QUAD;
AdDioSetup.IngrDioEngine[0].DmaBaseAddr = (Uint32)&dio_result[0];//DMA destination base address
(use high 28 bits)
AdDioSetup.IngrDioEngine[0].DmaBurstAddrStride = 4;// (wrap1)
AdDioSetup.IngrDioEngine[0].DmaBlockAddrStride = 6;// (wrap2)
AdDioSetup.IngrDioEngine[0].DBCN[0] = 0; //set ingress table DBCN for channel 0
AdDioSetup.IngrDioEngine[0].DBCN[1] = 1; //set ingress table DBCN for channel 1
AdDioSetup.IngrDioEngine[0].DBCN[2] = 2; //set ingress table DBCN for channel 2

//Ingress DIO Engine 1setup for RAC_B
AdDioSetup.IngrDioEngineEnable[1] = TRUE;//Enable DIO Engine 1
AdDioSetup.IngrDioEngine[1].BcnTableSelect = CSL_AIF2_AD_DIO_BCN_TABLE_0;
AdDioSetup.IngrDioEngine[1].NumQuadWord = CSL_AIF2_AD_2QUAD;
AdDioSetup.IngrDioEngine[1].NumAxC = 2;//Use 3 AxC. Set N-1
AdDioSetup.IngrDioEngine[1].bEnDmaChannel = TRUE; //Enable Dma channel
AdDioSetup.IngrDioEngine[1].DmaNumBlock = 3;//4 block wrap value (RAC loop) Set N-1
AdDioSetup.IngrDioEngine[1].DmaBurstLen = CSL_AIF2_AD_4QUAD;
AdDioSetup.IngrDioEngine[1].DmaBaseAddr = (Uint32)&dio_result1[0];//DMA destination base address
(use high 28 bits)
AdDioSetup.IngrDioEngine[1].DmaBurstAddrStride = 4;// (wrap1)
AdDioSetup.IngrDioEngine[1].DmaBlockAddrStride = 6;// (wrap2)
AdDioSetup.IngrDioEngine[1].DBCN[0] = 0; //set ingress table DBCN for channel 0
AdDioSetup.IngrDioEngine[1].DBCN[1] = 1; //set ingress table DBCN for channel 1
AdDioSetup.IngrDioEngine[1].DBCN[2] = 2; //set ingress table DBCN for channel 2

//AT In DIO Event 0 setup (In DIO 8chip Event)
AtEventSetup.AtIngrDioEvent[0].EventSelect = CSL_AIF2_IN_DIO_EVENT_0;//Select In DIO Event 0
AtEventSetup.AtIngrDioEvent[0].EventOffset = 0;//fine offset.
AtEventSetup.AtIngrDioEvent[0].EvtStrobeSel = CSL_AIF2_RADT_FRAME;
AtEventSetup.AtIngrDioEvent[0].EventModulo = 639; //WCDMA 8 chip time modulo
AtEventSetup.AtIngrDioEvent[0].DioFrameEventOffset = 1515;//Pi Max + 8 WCDMA chip time + PD delay
and fuzzy factors(190)
AtEventSetup.AtIngrDioEvent[0].DioFrameStrobeSel = CSL_AIF2_RADT_FRAME; //frame event strobe
selection
AtEventSetup.bEnableIngrDioEvent[0] = TRUE;//Enable In DIO Event 0

//AT In DIO Event 1 setup (In DIO 8chip Event)
AtEventSetup.AtIngrDioEvent[1].EventSelect = CSL_AIF2_IN_DIO_EVENT_1;//Select In DIO Event 1
AtEventSetup.AtIngrDioEvent[1].EventOffset = 0;//fine offset.
AtEventSetup.AtIngrDioEvent[1].EvtStrobeSel = CSL_AIF2_RADT_FRAME;
AtEventSetup.AtIngrDioEvent[1].EventModulo = 639; //WCDMA 8 chip time modulo
AtEventSetup.AtIngrDioEvent[1].DioFrameEventOffset = 1515;//Pi Max + 8 WCDMA chip time + PD delay
and fuzzy factors(190)
AtEventSetup.AtIngrDioEvent[1].DioFrameStrobeSel = CSL_AIF2_RADT_FRAME; //frame event strobe
selection
AtEventSetup.bEnableIngrDioEvent[1] = TRUE;//Enable In DIO Event 1
```

```
//AT Event 9 setup to generate 32 chip trigger for RAC_A (Event 9)
AtEventSetup.AtRadEvent[9].EventSelect = CSL_AIF2_EVENT_9;
AtEventSetup.AtRadEvent[9].EventOffset = 0;
AtEventSetup.AtRadEvent[9].EvtStrobeSel = CSL_AIF2_ULRADT_FRAME; //First Ul frame event will
occur after 4480 clocks
AtEventSetup.AtRadEvent[9].EventModulo = 2559; //WCDMA 32 chip time
AtEventSetup.AtRadEvent[9].EventMaskLsb = 0xFFFFFFFF;
AtEventSetup.AtRadEvent[9].EventMaskMsb = 0xFFFFFFFF;
AtEventSetup.bEnableRadEvent[9] = TRUE;//Enable Event 9

//AT Event 10 setup to generate 32 chip trigger for RAC_B (Event 10)
AtEventSetup.AtRadEvent[10].EventSelect = CSL_AIF2_EVENT_10;
AtEventSetup.AtRadEvent[10].EventOffset = 0;
AtEventSetup.AtRadEvent[10].EvtStrobeSel = CSL_AIF2_ULRADT_FRAME; //First Ul frame event will
occur after 4480 clocks
AtEventSetup.AtRadEvent[10].EventModulo = 2559; //WCDMA 32 chip time
AtEventSetup.AtRadEvent[10].EventMaskLsb = 0xFFFFFFFF;
AtEventSetup.AtRadEvent[10].EventMaskMsb = 0xFFFFFFFF;
AtEventSetup.bEnableRadEvent[10] = TRUE;//Enable Event 10
```

9.3.2 WCDMA Egress DIO Engine Setup For TAC 32 AxC Channels

```
//AD Common and DIO setup
AdCommonSetup.EgrGlobalEnable = TRUE;
AdCommonSetup.EgrGlobalDioEnable = TRUE;
AdCommonSetup.FailMode = CSL_AIF2_AD_DROP;//drop fail packet
AdCommonSetup.EgrPriority = CSL_AIF2_AD_AXC_PRI;
AdDioSetup.EgrDioEngineEnable[0] = TRUE;//Enable Egress DIO Engine
AdDioSetup.EgrDioEngine[0].BcnTableSelect = CSL_AIF2_AD_DIO_BCN_TABLE_0;
AdDioSetup.EgrDioEngine[0].NumQuadWord = CSL_AIF2_AD_1QUAD;
AdDioSetup.EgrDioEngine[0].NumAxC = 31;//Use 32 AxC
AdDioSetup.EgrDioEngine[0].bEnDmaChannel = TRUE; //Enable Dma channel
AdDioSetup.EgrDioEngine[0].bEnEgressRsaFormat = FALSE; //Disabled
AdDioSetup.EgrDioEngine[0].DmaNumBlock = 7;//8 block wrap value
AdDioSetup.EgrDioEngine[0].DmaBurstLen = CSL_AIF2_AD_4QUAD;//4 max QW burst per one transfer
AdDioSetup.EgrDioEngine[0].DmaBaseAddr =(Uint32)&dio_data[0];//DMA source base address (use high
28 bits)
AdDioSetup.EgrDioEngine[0].DmaBurstAddrStride = 4;// (wrap1)
AdDioSetup.EgrDioEngine[0].DmaBlockAddrStride = 0;// (wrap2)

for (i=0; i<32; i++)
AdDioSetup.EgrDioEngine[0].DBCN[i] = i; //set egress table DBCN for 32 channels

//AT Event setup (E DIO 4chip Event)
AtEventSetup.AtEgrDioEvent[0].EventSelect = CSL_AIF2_E_DIO_EVENT_0;//Select E DIO Event 0
AtEventSetup.AtEgrDioEvent[0].EventOffset = 300;//delay for TAC operation time
AtEventSetup.AtEgrDioEvent[0].EvtStrobeSel = CSL_AIF2_RADT_FRAME;
AtEventSetup.AtEgrDioEvent[0].EventModulo = 319; //set Modulus count for E DIO event 0 (WCDMA 4
chip time)
AtEventSetup.AtEgrDioEvent[0].DioFrameEventOffset = 0;
AtEventSetup.AtEgrDioEvent[0].DioFrameStrobeSel = CSL_AIF2_RADT_FRAME; //frame event strobe
selection
AtEventSetup.bEnableEgrDioEvent[0] = TRUE;//Enable E DIO Event 0

//AT Event setup for generating 4 chip trigger for TAC (Event 8 is specified for this purpose)
AtEventSetup.AtRadEvent[8].EventSelect = CSL_AIF2_EVENT_8;
AtEventSetup.AtRadEvent[8].EventOffset = 0;
AtEventSetup.AtRadEvent[8].EvtStrobeSel = CSL_AIF2_RADT_FRAME;
AtEventSetup.AtRadEvent[8].EventModulo = 319; //WCDMA 4 chip time
AtEventSetup.AtRadEvent[8].EventMaskLsb = 0xFFFFFFFF;
AtEventSetup.AtRadEvent[8].EventMaskMsb = 0xFFFFFFFF;
AtEventSetup.bEnableRadEvent[8] = TRUE;//Enable Event 8
```

9.4 AIF2 PKTDMA and QM Setup

9.4.1 LTE 4x, 1 AxC (Channel 0) With Protocol-Specific Field

```
//AIF2 Ingress DB setup
IngrDbSetup.bEnableIngrDb = TRUE; //Enable Ingress DB
IngrDbSetup.bEnableChannel[0] = TRUE; //Enable Ingress DB channel 0
IngrDbSetup.IngrDbChannel[0].BaseAddress = AIF2_DB_BASE_ADDR_I_FIFO_0;
IngrDbSetup.IngrDbChannel[0].BufDepth = CSL_AIF2_DB_FIFO_DEPTH_QW32; //Set DB FIFO depth for
channel 0 to 32 QWords
IngrDbSetup.IngrDbChannel[0].DataSwap = CSL_AIF2_DB_NO_SWAP;
IngrDbSetup.IngrDbChannel[0].IQOrder = CSL_AIF2_DB_IQ_NO_SWAP;
IngrDbSetup.IngrDbChannel[0].bEnablePsData = TRUE; //Enable 4 bytes PS data
IngrDbSetup.IngrDbChannel[0].PacketType = 0; //User defined value for packet type

//AIF2 Egress DB setup
EgrDbSetup.bEnableEgrDb = TRUE; //Enable Ingress DB
EgrDbSetup.PmControl = CSL_AIF2_DB_PM_TOKEN_FIFO;//for normal packet performance
EgrDbSetup.bEnableChannel[0] = TRUE; //Enable Egress DB channel 0
EgrDbSetup.EgrDbChannel[0].BaseAddress = AIF2_DB_BASE_ADDR_E_FIFO_0;
EgrDbSetup.EgrDbChannel[0].BufDepth = CSL_AIF2_DB_FIFO_DEPTH_QW32; //Set DB FIFO depth for
channel 0 to 32 QW
EgrDbSetup.EgrDbChannel[0].DataSwap = CSL_AIF2_DB_NO_SWAP;
EgrDbSetup.EgrDbChannel[0].IQOrder = CSL_AIF2_DB_IQ_NO_SWAP;

//AD Common setup
AdCommonSetup.IngrGlobalEnable = TRUE;
AdCommonSetup.EgrGlobalEnable = TRUE;
AdCommonSetup.FailMode = CSL_AIF2_AD_DROP;//drop fail packet
AdCommonSetup.IngrPriority = CSL_AIF2_AD_PKT_PRI;
AdCommonSetup.EgrPriority = CSL_AIF2_AD_AXC_PRI;
AdCommonSetup.Tx_QueNum = AIF2_BASE_TX_QUE_NUM;//base egress queue number setup to 512
//Queue Manager and AIF2 PKTDMA setup for LTE
```

9.4.1.1 Queue Manager Setup

Memory Region 0 setup for 32 * 8848-byte Monolithic descriptors. The Mono descriptors will be 12 bytes plus 4 bytes protocol specific field, plus 8832(8768) bytes of payload (symbol), so the total size is 8848 and it is dividable by 16. Assigned 32 descriptors because it is min descriptor value and named mono_region.

```
set_memory_region(0, (Uint32) mono_region, 0, 0x02280000);// region 0, index 0, setup is
0x02280000
set_memory_region function looks like below.
void set_memory_region(Uint16 regn, Uint32 addr,    Uint32 indx, Uint32 setup)
{
Uint32 *reg;
    reg = (Uint32 *)(QM_DESC_REGION + QM_REG_MEM_REGION_BASE + (regn * 16));
    *reg = addr;
    reg = (Uint32 *)(QM_DESC_REGION + QM_REG_MEM_REGION_INDEX + (regn * 16));
    *reg = indx;
    /* See register description for programming values. */
    reg = (Uint32 *)(QM_DESC_REGION + QM_REG_MEM_REGION_SETUP + (regn *    16));
    *reg = setup;
}

#define QMSS_CFG_BASE (0x02a00000u)
#define QM_DESC_REGION (QMSS_CFG_BASE + 0x0006a000u)
#define QM_REG_MEM_REGION_BASE 0x000
#define QM_REG_MEM_REGION_INDEX 0x004
#define QM_REG_MEM_REGION_SETUP 0x008
```

Next thing is configuring Linking RAM 0 for the descriptor regions.

```
set_link_ram(0, QM_LRAM_REGION, 0x3fff); //Link    ram0, internal link ram, 0x3fff number of
descriptors
set_link_ram function looks like below.
void set_link_ram(Uint16 ram, Uint32 addr, Uint32    count)
{
Uint32 *reg;
    reg = (Uint32 *)(QM_CTRL_REGION + QM_REG_LINKRAM_0_BASE + (ram * 8));
    *reg = addr;
    if (ram == 0)
    {
        reg = (Uint32 *)(QM_CTRL_REGION + QM_REG_LINKRAM_0_SIZE);
        *reg = count;
    }
}

#define QM_CTRL_REGION (QMSS_CFG_BASE + 0x00068000u)
#define QM_LRAM_REGION (0x00080000u)
#define QM_REG_LINKRAM_0_BASE 0x00C
```

Initialization of descriptor region 0 to zero

```
memset(mono_region, 0, 32 * 8848);
// Push 14 Monolithic symbol packets into Tx Completion Queue
for (idx = 0; idx < 14; idx ++)
{
    mono_pkt = (CPPI_MonolithicPacketDescriptor *)(mono_region + (idx * 8848));
    mono_pkt->type_id = CPPI_DESC_TYPE_MONO;
    mono_pkt->data_offset = 16; // 12 byte header + 4 byte PS field
    mono_pkt->pkt_return_qmgr = 0; //virtual queue manager number is zero
    mono_pkt->pkt_return_qnum = MONO_TX_COMPLETE_Q;
    push_queue(MONO_TX_COMPLETE_Q, 1, 0, (Uint32)(mono_pkt));
}

// Push 14 Monolithic packets to Rx FDQ
for (idx = 16; idx < 30; idx ++)
{
    mono_pkt = (CPPI_MonolithicPacketDescriptor *)(mono_region + (idx * 8848));
    mono_pkt->type_id = CPPI_DESC_TYPE_MONO;
    mono_pkt->data_offset = 16;
    push_queue(MONO_RX_FDQ, 1, 0, (Uint32)(mono_pkt));
}
```

"*push_queue*" and " *pop_quque*" functions look like below. Pop_queue function is used by application when pop a received queue for processing.

```
void push_queue(Uint16 qn, Uint8 mode, Uint32    c_val, Uint32 d_val)
{
Uint32 *reg;
    if (mode == 2)
    {
        reg = (Uint32 *)(QM_QMAN_REGION + QM_REG_QUE_REG_C + (qn * 16));
        *reg = c_val;
    }
    reg = (Uint32 *)(QM_QMAN_REGION + QM_REG_QUE_REG_D + (qn * 16));
    *reg = d_val;
}

Uint32 pop_queue(Uint16 qn)
{
    Uint32 *reg;
    Uint32 value;
    reg = (Uint32 *)(QM_QMAN_REGION + QM_REG_QUE_REG_D + (qn * 16));
    value = *reg;
    return(value);
}
```

```
#define QM_QMAN_REGION (QMSS_CFG_BASE + 0x00020000u)
#define QM_REG_QUE_REG_C 0x008
#define QM_REG_QUE_REG_D 0x00c
// Queue definition
#define MONO_TX_COMPLETE_Q 2000
#define MONO_RX_FDQ 2001
#define MONO_RX_Q 900
#define MONO_TX_Q 512
```

9.4.1.2 AIF2 PKTDMA Module Setup

Configure Rx channel flows

```
//Create flow configuration 0 for the Monolithic packets
flow_a = 0x28100000 | MONO_RX_Q;
flow_d = MONO_RX_FDQ << 16;
config_rx_flow(AIF_CDMA_RX_FLOW_REGION, 0,
flow_a, 0, 0, flow_d, 0, 0, 0, 0);
```

" *config_rx_flow*" function looks like below

```
void config_rx_flow(Uint32 base, Uint16    flow,
Uint32 a, Uint32 b, Uint32 c, Uint32 d,
Uint32 e, Uint32 f, Uint32 g, Uint32 h)
{
    Uint32 *reg;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_A + (flow * 32));
    *reg = a;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_B + (flow * 32));
    *reg = b;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_C + (flow * 32));
    *reg = c;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_D + (flow * 32));
    *reg = d;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_E + (flow * 32));
    *reg = e;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_F + (flow * 32));
    *reg = f;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_G + (flow * 32));
    *reg = g;
    reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_H + (flow * 32));
    *reg = h;
}

#define AIF_CDMA_RX_FLOW_REGION (AIF_CFG_BASE + 0x0001a000u)
#define CDMA_REG_RX_FLOW_CFG_A 0x000
#define CDMA_REG_RX_FLOW_CFG_B 0x004
#define CDMA_REG_RX_FLOW_CFG_C 0x008
#define CDMA_REG_RX_FLOW_CFG_D 0x00c
#define CDMA_REG_RX_FLOW_CFG_E 0x010
#define CDMA_REG_RX_FLOW_CFG_F 0x014
#define CDMA_REG_RX_FLOW_CFG_G 0x018
#define CDMA_REG_RX_FLOW_CFG_H 0x01c
```

Enable Tx and Rx channels and disable PKTDMA loopback mode and set AIF_MONO_MODE for LTE usage.

```
enable_disable_loopback(0);//disable PKTDMA    loopback for normal data transfer
config_tx_chan(AIF_CDMA_TX_CHAN_REGION, 0,    0x01000000); //set AIF_MONO_MODE to 1 and set PS
filter to zero
enable_tx_chan(AIF_CDMA_TX_CHAN_REGION, 0,    0x80000000);
enable_rx_chan(AIF_CDMA_RX_CHAN_REGION, 0,    0x80000000);
```

functions used above defined below.

```
void enable_disable_loopback(Uint32 value)
{
    Uint32 *reg;
    reg = (Uint32 *)(AIF_CDMA_GBL_CFG_REGION + 0x04);
    *reg = 0x00080000; //set write arbitration FIFO depth to 8
    reg = (Uint32 *)(AIF_CDMA_GBL_CFG_REGION + 0x08);
    *reg = value;
}

void config_tx_chan(Uint32 base, Uint16 chan,    Uint32 reg_b)
{
    Uint32 *reg;
    reg = (Uint32 *)(base + CDMA_REG_TX_CHAN_CFG_A + (chan * 32));
    *reg = 0; //disable the channel
    reg = (Uint32 *)(base + CDMA_REG_TX_CHAN_CFG_B + (chan * 32));
    *reg = reg_b;
}

void enable_rx_chan(Uint32 base, Uint16 chan,    Uint32 value)
{
    Uint32 *reg;
    reg = (Uint32 *)(base + CDMA_REG_RX_CHAN_CFG_A + (chan * 32));
    *reg = value;
}

void enable_tx_chan(Uint32 base, Uint16 chan,    Uint32 value)
{
    Uint32 *reg;
    reg = (Uint32 *)(base + CDMA_REG_TX_CHAN_CFG_A + (chan * 32));
    *reg = value;
}

#define AIF_CDMA_GBL_CFG_REGION (AIF_CFG_BASE + 0x00014000u)
#define AIF_CDMA_TX_CHAN_REGION (AIF_CFG_BASE + 0x00016000u)
#define AIF_CDMA_RX_CHAN_REGION (AIF_CFG_BASE + 0x00018000u)
#define CDMA_REG_TX_CHAN_CFG_A 0x000
#define CDMA_REG_TX_CHAN_CFG_B 0x004
#define CDMA_REG_RX_CHAN_CFG_A 0x000
```

9.4.1.3 Tx Queue Push Routine

To start Egress symbol data transfer, the application software should create contents of the descriptors and push packets into the Tx queue. If FFTC does this job, the descriptor contents should be created before processing data through the FFTC. The following code snippet shows how to create and push packets into the Tx queue manually.

```
for (idx =0; idx < 14; idx++){ //push 14 LTE symbol packets into Tx queue for    test
tmp = pop_queue(MONO_TX_COMPLETE_Q);
tmp &= 0xFFFFFFF0;//set DESC_SIZE field to zero
mono_pkt = (CPPI_MonolithicPacketDescriptor *)tmp;

//Create Mono packet (initialize non-zero fields)
mono_pkt->type_id = CPPI_DESC_TYPE_MONO;
mono_pkt->data_offset = CPPI_MONO_PACKET_SIZE + 4;//16 bytes
if((idx == 0) || (idx == 7))
mono_pkt->packet_length = 8832; //first symbol
else mono_pkt->packet_length = 8768; //other six symbols
mono_pkt->ps_flags = 0;
mono_pkt->epib = 0;
mono_pkt->psv_word_count = 1; // 4 byte PS field length
mono_pkt->pkt_return_qnum = MONO_TX_COMPLETE_Q;
mono_pkt->src_tag_lo = 0; //copied to .flo_idx of streaming i/f
temp = (Uint32 *)(tmp + 16);
if(idx == 0){
    for (idx2 = 0; idx2 < 2208; idx2 ++) temp[idx2] = idx2; //payload data setup(first symbol)
}
else {
    for (idx2 = 0; idx2 < 2192; idx2 ++) temp[idx2] = idx2; //payload data    setup(other six
symbols)
}

//Create PS data
temp = (Uint32 *)(tmp + CPPI_MONO_PACKET_SIZE);
temp[0] = (Uint32)(0x00008000 + (idx << 7)); //add symbol number into PS    field
tmp |= 0x00000003; //set DESC_SIZE to 3 for AIF2 mono mode
push_queue(MONO_TX_Q, 1, 0, tmp);
}

#define CPPI_MONO_PACKET_SIZE 12
typedef struct
{

/* word 0 */
Uint32 type_id : 2; //always 0x2 (Monolithic Packet ID)
Uint32 packet_type : 5;
Uint32 data_offset : 9;
Uint32 packet_length : 16; //in bytes (65535 max)

/* word 1 */
Uint32 src_tag_hi : 8;
Uint32 src_tag_lo : 8;
Uint32 dest_tag_hi : 8;
Uint32 dest_tag_lo : 8;

/* word 2 */
Uint32 epib : 1; //1=extended packet info block is present
Uint32 reserved_w2 : 1;
Uint32 psv_word_count : 6; //number of 32-bit PS data words
Uint32 err_flags : 4;
Uint32 ps_flags : 4;
Uint32 reserved_w2b : 1;
Uint32 on_chip : 1; //0=descriptor is in off-chip, 1=on-chip
Uint32 pkt_return_qmgr : 2;
Uint32 pkt_return_qnum : 12;
} CPPI_MonolithicPacketDescriptor;
```

9.5 Generic Packet Traffic Setup

AIF2 mainly supports AxC data (I,Q data for Radio) traffic and Generic packet traffic.

For AxC data, the packet or symbol boundary is decided by counting radio framing counters like RadTimerTc, FrameSymbolTC, FrameMsgTc, and even AxC offset in the PD, PE module; but those schemes are not used for generic packet traffic. Instead, a special packet-framing scheme could be used to differentiate the boundary of each packet. For OBSAI, we use Timestamp field in OBSAI header and use 4B/5B encoding, Null delimiter, or, in the case of CPRI, the Hyper frame boundary.

Radio framing requires correct timing between symbols and only well-known symbol size could be used to easily predict the boundary of symbols within contiguous radio frames but packet framing works in a totally different way. It does not require tight and correct packet timing and packet size; the packet data format is also flexible and it is totally dependant upon user application.

AIF2 packet mode supports the following generic packet modes:

- CPRI Ethernet
- CPRI Control Data
- OBSAI 16 byte Generic Control Message
- OBSAI or CPRI Generic Packet
- OBSAI or CPRI Generic usage of AxC Data Payload

The Generic packet traffic mode is envisioned to support all packet switch messages (including control messages) and is used to facilitate inter-DSP communication traffic. The Generic format is differentiated from other formats with its word (4 byte) orientation matching the width of the DSP core data path. The example in this chapter shows how to setup Packet framing with OBSAI and CPRI Generic Packet format.

9.5.1 OBSAI Generic Packet Traffic

Generic Packet is an OBSAI mechanism which allows multiple OBSAI messages to be grouped together to form a larger packet. The packet size is n x 16 bytes where the last two bytes are the CRC16. Generic Packet type fields should be always 5b01111 and the timestamp field looks like below.

Timestamp Field

- Bit[5:4]
  - 2'b10: SOP start
  - 2'b00: MOP mid-point
  - 2'b11: EOP end
- Bit[3:0]: extension of address field (if needed)

OBSAI has AxC slot and Control slot (20:1 ratio) and both type of slot could be used to transfer generic packet. Below example only uses AxC slots for generic packet traffic and shows how to configure AIF2 OBSAI link and common modules.

```
//Link Common setup
ComLinkSetup.linkProtocol = CSL_AIF2_LINK_PROTOCOL_OBSAI;
ComLinkSetup.linkRate = CSL_AIF2_LINK_RATE_4x; // 4x or 8x link speed is good for generic packet
transfer
ComLinkSetup.IngrDataWidth = CSL_AIF2_DATA_WIDTH_16_BIT; // should be 16 bit width
ComLinkSetup.EgrDataWidth = CSL_AIF2_DATA_WIDTH_16_BIT;
SD, RM, TM, AT module setup is same to Normal AxC data case.

//RT link setup
RtLinkSetup.bEnableEmptyMsg = TRUE;
RtLinkSetup.RtConfig = CSL_AIF2_RT_MODE_TRANSMIT;// should select PE only mode

//PD link setup
PdLinkSetup.PdTypeLut[15].ObsaiTsFormat = CSL_AIF2_TSTAMP_FORMAT_GEN_PKT;
PdLinkSetup.PdTypeLut[15].PdCrcType = CSL_AIF2_CRC_16BIT;
PdLinkSetup.PdTypeLut[15].bEnableCrc = TRUE;
PdLinkSetup.PdTypeLut[15].PdObsaiMode = CSL_AIF2_PD_DATA_PKT;
PdLinkSetup.PdTypeLut[15].bEnableEnetStrip = FALSE;
PdLinkSetup.PdTypeLut[15].bEnableCrcHeader = FALSE;

//PE link setup
PeLinkSetup.PeCppiDioSel = CSL_AIF2_CPPI;
PeLinkSetup.TddAxc = FALSE;
PeLinkSetup.bEnObsaiBubbleBW = FALSE;
PeLinkSetup.PeDelay = DB_PE_DELAY_OBSAI;//28 sys_clks delay between DB and PE for OBSAI
```

In PD, PE common setup, AxC offset window, AxC offset, RadtTC, FrameTC and FrameMsgTC is not used for generic packet mode. In case of AxC data transfer, we had to keep correct radio standard timing and AxC data on/off time. If we lose the correct channel timing, PD (PE) doesn't work for the whole radio frame time. But in case of generic packet mode, we don't need to keep that correct timing. User program can send any size of packet at any time they want. PD and PE don't really care the junk duration between packets and fills empty message during that time. Timestamp field shows which message has SOP or EOP, so PD easily detect which OBSAI message has valid data or invalid data.

```
//PD common setup
PdCommonSetup.PdCppiDioSel = CSL_AIF2_CPPI;
PdCommonSetup.PdRoute[0].RouteTs = 0x0;//Route OBSAI timestamp for channel 0
PdCommonSetup.PdRoute[0].RouteType = OBSAI_TYPE_GENERIC;//Route OBSAI type for channel 0
PdCommonSetup.PdRoute[0].RouteAddr = 0;//Route OBSAI address for channel 0
PdCommonSetup.PdRoute[0].RouteLink = CSL_AIF2_LINK_0;//Route link for channel 0
PdCommonSetup.PdRoute[0].RouteMask = CSL_AIF2_ROUTE_MASK_4LSB;//Route TS mask for channel 0
PdCommonSetup.PdChConfig[0].bChannelEn = TRUE;//Channel enable for channel 0
PdCommonSetup.PdChConfig[0].DataFormat = CSL_AIF2_LINK_DATA_TYPE_NORMAL;
PdCommonSetup.PdChConfig1[0].bTsWatchDogEn = FALSE;//disable watchdog for channel 0
PdCommonSetup.PdChConfig1[0].DataFormat = CSL_AIF2_GSM_DATA_OTHER;//Non GSM data
PdCommonSetup.PdChConfig1[0].TddEnable = 0xFFFF;//PD TDD, enables all symbol for FDD for channel
0
PdCommonSetup.TddEnable1[0] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable2[0] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable3[0] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable4[0] = 0xFFFFFFFF;//enables all symbols(FDD)

//PE common setup
PeCommonSetup.PeTokenPhase = 1;//Phase alignment for scheduling DMA OBSAI: only lsb is used
PeCommonSetup.EnetHeaderSelect = 1;//bit order for Ethernet preamble and SOF
PeCommonSetup.bEnableCh[0] = TRUE;//Enable PE channel for channel 0
PeCommonSetup.PeDmaCh0[0].bCrcEn = TRUE;// enable CRC 16 for channel 0
PeCommonSetup.PeDmaCh0[0].RtControl = CSL_AIF2_PE_RT_INSERT;//use PE insert option for channel 0
PeCommonSetup.PeDmaCh0[0].CrcType = CSL_AIF2_CRC_16BIT;//CRC type for channel 0
PeCommonSetup.PeDmaCh0[0].isEthernet = FALSE;
PeCommonSetup.PeDmaCh0[0].CrcObsaiHeader = FALSE;
PeCommonSetup.PeInFifo[0].SyncSymbol = 0;//sync symbol offset for channel 0
PeCommonSetup.PeInFifo[0].MFifoWmark = 2;//Message FIFO water mark
PeCommonSetup.PeInFifo[0].MFifoFullLevel = 3;//Message FIFO full level
PeCommonSetup.PeModuloTc[0].bEnableRule = TRUE;
PeCommonSetup.PeModuloTc[0].RuleModulo = 0;//Setup modulo rule 0 Modulo
```

```
PeCommonSetup.PeModuloTc[0].bRuleObsaiCtlMsg = FALSE; //use normal payload
PeCommonSetup.PeModuloTc[0].RuleIndex = 0;//Setup modulo rule 0 index
PeCommonSetup.PeModuloTc[0].RuleLink = CSL_AIF2_LINK_0;//Route egress modulo rule 0 to link 0
PeCommonSetup.PeChObsaiType[0] = OBSAI_TYPE_GENERIC;//OBSAI header type for channel 0
PeCommonSetup.PeChObsaiTS[0] = 0x0;//OBSAI header Timestamp for channel 0
PeCommonSetup.PeChObsaiAddr[0] = 0;//OBSAI header address for channel 0
PeCommonSetup.PeChObsaiTsMask[0] = CSL_AIF2_ROUTE_MASK_4LSB;
PeCommonSetup.PeChObsaiTsfomat[0] = CSL_AIF2_TSTAMP_FORMAT_GEN_PKT;
PeCommonSetup.PeObsaiPkt[0] = TRUE;//Select OBSAI AxC or packet mode for channel 0
PeCommonSetup.PeBbHop[0] = FALSE;//Take OBSAI address from Multicore Navigator PS bits for
channel 0
For DBMR setup, users can choose the X number for their needs. X=1 could be used to get maximum
data bandwidth for one DB channel. If user wants to split whole bandwidth into multiple DB
channels, multiple number of X could be used and Multicore Navigator needs to be set to control
each channel correctly.

//Dual bit map setup.
PeCommonSetup.PeObsaiDualBitMap[0].DbmX = 0;//set X-1. use max bandwidth for one channel
PeCommonSetup.PeObsaiDualBitMap[0].DbmXBubble = 0;
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Mult = 0;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Size = 0;//set n-1
PeCommonSetup.PeObsaiDualBitMap[0].Dbm1Map[0] = 0x0;
PeCommonSetup.ChIndex0[0] = 0; //channel 0
PeCommonSetup.bEnableChIndex0[0] = TRUE;//Route egress channel 0 to modulo rule 0

//Ingress DB setup
IngrDbSetup.bEnableIngrDb = TRUE; //Enable Ingress DB
IngrDbSetup.bEnableChannel[0] = TRUE; //Enable Ingress DB channel 0
IngrDbSetup.IngrDbChannel[0].BaseAddress = AIF2_DB_BASE_ADDR_I_FIFO_0;
IngrDbSetup.IngrDbChannel[0].BufDepth = CSL_AIF2_DB_FIFO_DEPTH_QW64; // 128 QW is also possible
IngrDbSetup.IngrDbChannel[0].bEnablePsData = FALSE; //Disable PS data
IngrDbSetup.IngrDbChannel[0].PacketType = 0; //User data

//Egress DB setup
EgrDbSetup.bEnableEgrDb = TRUE; //Enable Ingress DB
EgrDbSetup.PmControl = CSL_AIF2_DB_PM_TOKEN_FIFO;//for normal packet performance
EgrDbSetup.bEnableChannel[0] = TRUE; //Enable Egress DB channel 0
EgrDbSetup.EgrDbChannel[0].BaseAddress = AIF2_DB_BASE_ADDR_E_FIFO_0;
EgrDbSetup.EgrDbChannel[0].BufDepth = CSL_AIF2_DB_FIFO_DEPTH_QW128; // 64QW is also possible

//AD Common setup
AdCommonSetup.IngrGlobalEnable = TRUE;
AdCommonSetup.EgrGlobalEnable = TRUE;
AdCommonSetup.FailMode = CSL_AIF2_AD_DROP;//drop fail packet
AdCommonSetup.IngrPriority = CSL_AIF2_AD_PKT_PRI;
AdCommonSetup.EgrPriority = CSL_AIF2_AD_NON_AXC_PRI;
AdCommonSetup.Tx_QueNum = AIF2_BASE_TX_QUE_NUM;//base egress queue number setup to 512
```

For periodic packet transfer, use the AT external event to push and pop packet descriptors. For non-periodic packet transfer, Multicore Navigator accumulation could be used to issue an interrupt for CorePac application. The *Multicore Navigator for Keystone Devices User Guide* (see) has more information about accumulator operation and configuration.

9.5.2 CPRI Generic Packet Traffic

The OBSAI standard has specified very powerful packet-switched traffic approach. AIF2 fully supports these OBSAI requirements and overlays some of the supporting hardware with CPRI usage. OBSAI allows for very high bandwidth by allowing unused AxC slots to pass generic packet or control data. OBSAI also allows for great flexibility in number of supported control streams and placement of control streams within the link packing but in case of CPRI, there are some restrictions when compared to OBSAI. Main features of CPRI are:

- CPRI also allows very high bandwidth by allowing AxC slots to pass Generic packet or Ethernet data.
- CPRI only allows four DB channels to be used for Generic packet, Control data and Ethernet. Packet stream 0 ~ 3 can be matched with any four of DB 128 channels per link.
- Packets contained in CPRI AxC slot must be n * 16 bytes and no fragmented quad word supported on DMA side.
- Minimum CPRI packet size is 4 byte and CRC16 is used for generic packet mode in AxC slot.
- Only 4B/5B packet framing scheme is supported for packets in CPRI AxC slot and can not use Null delimiter (Control slots are fine for both schemes)

Below example only uses AxC slot for generic packet traffic and shows how to configure AIF2 CPRI link and common modules.

```
//Link Common setup
ComLinkSetup.linkProtocol = CSL_AIF2_LINK_PROTOCOL_CPRI;
ComLinkSetup.linkRate = CSL_AIF2_LINK_RATE_4x;
ComLinkSetup.IngrDataWidth = CSL_AIF2_DATA_WIDTH_16_BIT; // only use 4 byte sample size
ComLinkSetup.EgrDataWidth = CSL_AIF2_DATA_WIDTH_16_BIT;

//PD link setup
PdLinkSetup.CpriEnetStrip = 0;//disable ethernet strip for control channel
PdLinkSetup.CpriCwPktDelimitor[0] = CSL_AIF2_CW_DELIM_4B5B;
PdLinkSetup.PdCpriCrcType[0] = CSL_AIF2_CRC_16BIT;
PdLinkSetup.bEnableCpriCrc[0] = TRUE;//enable CPRI CRC for control channel 0
PdLinkSetup.PdPackDmaCh[0] = 0;//Set DB channel 0 as a dma ch for control channel 0
PdLinkSetup.bEnablePack[0] = TRUE;//enable CPRI control channel 0 packing
PdLinkSetup.PdCpriDualBitMap.DbmX = 0;// set X-1. Use Max bandwidth for DB channel 0
PdLinkSetup.PdCpriDualBitMap.DbmXBubble = 1;//2 bubbles of 1 AxC sample
PdLinkSetup.PdCpriDualBitMap.Dbm1Mult = 0;//set n-1
PdLinkSetup.PdCpriDualBitMap.Dbm1Size = 0;//set n-1
PdLinkSetup.PdCpriDualBitMap.Dbm1Map[0] = 0x0;
PdLinkSetup.PdCpriDualBitMap.Dbm2Size = 0;
PdLinkSetup.PdCpriDualBitMap.Dbm2Map[0] = 0x0;
PdLinkSetup.CpriDmaCh[0]= 0; //match DbmX channel 0 to DB channel 0
PdLinkSetup.bEnableCpriX[0]= TRUE; //enable CPRI X channel 0
PdLinkSetup.bEnableCpriPkt[0]= TRUE;//use Packet data for X channel 0
PdLinkSetup.Cpri8WordOffset[0]= 0;//fine CPRI AxC offset for X channel 0

//PE link setup
PeLinkSetup.PeCppiDioSel = CSL_AIF2_CPPI;
PeLinkSetup.TddAxc = FALSE;
PeLinkSetup.PeDelay = DB_PE_DELAY_CPRI;//0 sys_clks delay between DB and PE
PeLinkSetup.PeCpriDualBitMap.DbmX = 0;//set X-1
PeLinkSetup.PeCpriDualBitMap.DbmXBubble = 1;//2 bubbles of 1 AxC sample
PeLinkSetup.PeCpriDualBitMap.Dbm1Mult = 0;//set n-1
PeLinkSetup.PeCpriDualBitMap.Dbm1Size = 0;//set n-1
PeLinkSetup.PeCpriDualBitMap.Dbm1Map[0] = 0x0;
PeLinkSetup.PeCpriDualBitMap.Dbm2Size = 0;
PeLinkSetup.PeCpriDualBitMap.Dbm2Map[0] = 0x0;
PeLinkSetup.CpriAxCPack = CSL_AIF2_CPRI_16BIT_SAMPLE;
PeLinkSetup.CpriCwPktDelimitor[0] = CSL_AIF2_CW_DELIM_4B5B;
PeLinkSetup.PePackDmaCh[0] = 0;
PeLinkSetup.bEnablePack[0] = TRUE;

//PD common setup
PdCommonSetup.PdCppiDioSel = CSL_AIF2_CPPI;
PdCommonSetup.PdChConfig[0].bChannelEn = TRUE;//enable channel 0
```

```
PdCommonSetup.PdChConfig[0].DataFormat = CSL_AIF2_LINK_DATA_TYPE_NORMAL;
PdCommonSetup.PdChConfig1[0].DataFormat = CSL_AIF2_GSM_DATA_OTHER;//Non GSM data
PdCommonSetup.PdChConfig1[0].TddEnable = 0xFFFF;//PD TDD, enables all symbols(FDD) for channel 0
PdCommonSetup.TddEnable1[0] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable2[0] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable3[0] = 0xFFFFFFFF;//enables all symbols(FDD)
PdCommonSetup.TddEnable4[0] = 0xFFFFFFFF;//enables all symbols(FDD)

//PE common setup
PeCommonSetup.PeTokenPhase = 0;
PeCommonSetup.EnetHeaderSelect = 1;
PeCommonSetup.bEnableCh[0] = TRUE;//Enable PE channel for channel 0
PeCommonSetup.PeDmaCh0[0].bCrcEn = TRUE;//enable CRC for channel 0
PeCommonSetup.PeDmaCh0[0].RtControl = CSL_AIF2_PE_RT_INSERT;//use PE insert option for channel 0
PeCommonSetup.PeDmaCh0[0].CrcType = CSL_AIF2_CRC_16BIT;//CRC type for channel 0
PeCommonSetup.PeDmaCh0[0].isEthernet = FALSE;//AxC data
PeCommonSetup.PeInFifo[0].MFifoWmark = 2;//Message FIFO water mark
PeCommonSetup.PeInFifo[0].MFifoFullLevel = 3;//Message FIFO full level
PeCommonSetup.PeInFifo[0].SyncSymbol = 0;

//PE Channel LUT setup and link routing selection (ChIndex number is matched with link number)
PeCommonSetup.ChIndex2[0] = 0; //channel 0 for link 2
PeCommonSetup.bEnableChIndex2[0] = TRUE;//Route egress channel 0 to dedicated CPRI link
PeCommonSetup.CpriPktEn2[0] = TRUE; //use channel 0 for Packet

//Egress DB setup
EgrDbSetup.bEnableEgrDb = TRUE; //Enable Ingress DB
EgrDbSetup.PmControl = CSL_AIF2_DB_AXC_TOKEN_FIFO; // to enhance CPRI performance
EgrDbSetup.bEnableChannel[0] = TRUE; //Enable Egress DB channel 0
EgrDbSetup.EgrDbChannel[0].BaseAddress = AIF2_DB_BASE_ADDR_E_FIFO_0;
EgrDbSetup.EgrDbChannel[0].BufDepth = CSL_AIF2_DB_FIFO_DEPTH_QW128;
CPRI Multicore Navigator usage is the same to OBSAI case.
```

Revision History

Changes from October 1, 2012 to February 6, 2015	Page
• Corrected register name to pe_cw_lut.	315
• Added pe_global_en_clr register.	327

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265