Product IP

Internal Name: GEM_GXL
Part Number: IP7014

Gigabit Ethernet MAC with DMA, 1588, AVB and PCS (GEM_GXL)

User Guide

Revision 15

7 April 2016

Cadence Confidential

Document Number: I-IPA01-0266-USR Rev 15

Confidentiality Notice

Table of contents

Table of contents iii
Section 1 – Module Data 1
- Scope 1
- MAC Features 2
- Optional DMA Features 3
- Description 4
  - MAC 4
  - PCS 5
  - REG_TOP 5
  - DMA_TOP 5
  - TSU 6
- Block Diagram 6
- Signal Interfaces 9
  - Ethernet Interface (GMII/MII) 10
  - Ethernet Interface (RGMII) 10
  - Ethernet Interface (RMII) 11
  - MDIO Interface 11
  - Control/Status Interface 12
  - External Filter Interface 14
  - IEEE 1588 PTP frame recognition and Time Stamp Unit 15
  - Ethernet Interface (TBI) 17
  - Interrupt Controller Interface 18
  - External Transmit FIFO Interface 18
  - External Transmit FIFO Interface Status 19
  - External Receive FIFO Interface 20
  - AMBA (APB) Interface 22
  - AMBA (AHB) Interface 23
  - AMBA (AXI4) Interface 24
  - Transmit Packet Buffer Memory Interface (Dual Port SRAM Configuration) 26
  - Transmit Packet Buffer Memory Interface (Single Port SRAM Configuration) 26
  - Receive Packet Buffer Memory Interface (Dual Port SRAM Configuration) 27
  - Receive Packet Buffer Memory Interface (Single Port SRAM Configuration) 27
  - System Interface 29
- Timing Requirements 29
  - FIFO Interface Timing 30
  - Transmit Timing (MII/GMII) 31
  - Transmit Timing (RGMII) 31
  - Transmit Timing (RMII) 31
  - Receive Timing (MII/GMII) 32
  - Receive Timing (RGMII) 32
  - Receive Timing (RMII) 33
  - Transmit Timing (TBI) 33
  - Receive Timing (TBI) 33
  - Packet Buffer Memory Interface Timing 34
  - Clock Domains 34
- Programming Interface 36
  - Register Map 37
  - Physical Estimates 45

Section 2 - Programming the GEM_GXL 46
- Introduction 46
- Functional Description and Operation 47
  - MAC Functional Block Diagram 47
  - PCS Functional Block Diagram 48
  - Direct Memory Access Interface 48
    - Internal transmit/receive FIFO DMA 48

Packet Buffer DMA 49
Using the AXI Interface 49
Partial Store and Forward Using Packet Buffer DMA 50
DMA Transactions 51
Locked Accesses when using AHB and AHB Bus Ownership 51
Receive DMA Buffers 52
Transmit Buffers 58
DMA Bursting 65
DMA Endianism 66
DMA Transmit and Receive FIFOs 66
DMA Packet Buffer 67
External Memories: 68
Determining RAM Size 69
Reducing RAM Size 69
Reducing RAM Size – Transmit Operation 69
Reducing RAM Size – Partial Store and Forward Mode 70
Dual Port External Memory: 70
Single Port External Memory: 71
Transmit Packet Buffer: 71
Receive Packet Buffer: 72
Priority Queuing in the DMA 73
Advanced DMA Features to Reduce CPU Overhead 76
Receive Header/Data Splitting 76
Receive Side Coalescing 76
Large Send Offload 79
TSO Operation 79
UFO Operation 80
External FIFO Interface 81
Description 81
Priority Queuing without the DMA 82
External FIFO Interface Timing Criteria 84
Additional Low Latency TX FIFO Interface for DMA configurations 86
Transmit Scheduling Algorithm 87
MAC Transmit Block 89
MAC Receive Block 90
Checksum Offload for IP, TCP and UDP 91
Receiver Checksum Offload 91
Transmitter Checksum Offload 92
MAC Filtering Block 92
Broadcast Address 94
Hash Addressing 94
External Address Matching 94
Copy All Frames (or Promiscuous Mode) 96
Disable Copy of Pause Frames 96
VLAN Support 96
Wake on LAN Support 96
IEEE 1588 and IEEE 802.1AS Support 97
Support for Timestamping and Timestamp Accuracy 102
MAC 802.3 Pause Frame Support 105
802.3 Pause Frame Reception 106
802.3 Pause Frame Transmission 106
MAC PFC Priority Based Pause Frame Support 107
PFC Pause Frame Reception 107
PFC Pause Frame Transmission 108
Energy Efficient Ethernet support 109
LPI Operation in Cadence IP 110
PHY Interface 110
10/100/1000 Operation 111
SGMII Operation 111
Jumbo Frames 111

Physical Coding Sub-Layer 112
PCS Transmit 112
PCS Receiver 112
PCS Auto-negotiation 113
PCS Collision Detect and Carrier Sense 114
Link Status 114
Programming Interface 114
Registers 115
network_control 115
network_config 118
network_status 121
user_io_register 122
dma_config 123
transmit_status 126
receive_q_ptr 127
transmit_q_ptr 128
receive_status 129
int_status 130
int_enable 133
int_disable 135
int_mask 137
phy_management 141
pause_time 141
tx_pause_quantum 142
pbuf_txcutthru 142
pbuf_rxcutthru 142
jumbo_max_length 143
external_fifo_interface 143
axi_max_pipeline 143
rsc_control 144
int_moderation 144
sys_wake_time 145
hash_bottom 145
hash_top 145
spec_add1_bottom 146
spec_add1_top 146
spec_add2_bottom 146
spec_add2_top 147
spec_add3_bottom 147
spec_add3_top 148
spec_add4_bottom 148
spec_add4_top 149
spec_type1 149
spec_type2 149
spec_type3 150
spec_type4 150
wol_register 151
stretch_ratio 151
stacked_vlan 152
tx_pfc_pause 152
mask_add1_bottom 152
mask_add1_top 153
dma_addr_or_mask 153
rx_ptp_unicast 154
tx_ptp_unicast 154
tsu_nsec_cmp 154
tsu_sec_cmp 154
tsu_msb_sec_cmp 155
tsu_ptp_tx_msb_sec 155
tsu_ptp_rx_msb_sec 155

tsu_peer_tx_msb_sec 156
tsu_peer_rx_msb_sec 156
dpram_fill_dbg 157
revision_reg 157
octets_txed_bottom 158
octets_txed_top 158
frames_txed_ok 159
broadcast_txed 159
multicast_txed 159
pause_frames_txed 160
frames_txed_64 160
frames_txed_65 160
frames_txed_128 161
frames_txed_256 161
frames_txed_512 161
frames_txed_1024 161
frames_txed_1519 162
tx_underruns 162
single_collisions 162
multiple_collisions 163
excessive_collisions 163
late_collisions 163
deferred_frames 164
crs_errors 164
octets_rxed_bottom 164
octets_rxed_top 165
frames_rxed_ok 165
broadcast_rxed 165
multicast_rxed 166
pause_frames_rxed 166
frames_rxed_64 166
frames_rxed_65 167
frames_rxed_128 167
frames_rxed_256 167
frames_rxed_512 168
frames_rxed_1024 168
frames_rxed_1519 168
undersize_frames 169
excessive_rx_length 169
rx_jabbers 169
fcs_errors 170
rx_length_errors 170
rx_symbol_errors 171
alignment_errors 171
rx_resource_errors 172
rx_overruns 172
rx_ip_ck_errors 172
rx_tcp_ck_errors 173
rx_udp_ck_errors 173
auto_flushed_pkts 174
tsu_timer_incr_sub_nsec 174
tsu_timer_msb_sec 175
tsu_strobe_msb_sec 175
tsu_strobe_sec 175
tsu_strobe_nsec 176
tsu_timer_sec 176
tsu_timer_nsec 176
tsu_timer_adjust 177
tsu_timer_incr 177
tsu_ptp_tx_sec 178

tsu_ptp_tx_nsec 178
tsu_ptp_rx_sec 178
tsu_ptp_rx_nsec 179
tsu_peer_tx_sec 179
tsu_peer_tx_nsec 179
tsu_peer_rx_sec 180
tsu_peer_rx_nsec 180
pcs_control 181
pcs_status 182
pcs_an_adv 183
pcs_an_lp_base 184
pcs_an_exp 185
pcs_an_np_tx 186
pcs_an_lp_np 187
pcs_an_ext_status 187
tx_pause_quantum1 188
tx_pause_quantum2 188
tx_pause_quantum3 188
rx_lpi 189
rx_lpi_time 189
tx_lpi 189
tx_lpi_time 190
designcfg_debug1 190
designcfg_debug2 191
designcfg_debug3 191
designcfg_debug4 191
designcfg_debug5 192
designcfg_debug6 193
designcfg_debug7 194
designcfg_debug8 194
designcfg_debug9 195
designcfg_debug10 196
spec_add5_bottom 196
spec_add5_top 197
spec_add6_bottom 197
spec_add6_top 198
spec_add7_bottom 198
spec_add7_top 199
spec_add8_bottom 199
spec_add8_top 200
spec_add9_bottom 200
spec_add9_top 201
spec_add10_bottom 201
spec_add10_top 202
spec_add11_bottom 202
spec_add11_top 203
spec_add12_bottom 203
spec_add12_top 204
spec_add13_bottom 204
spec_add13_top 205
spec_add14_bottom 205
spec_add14_top 206
spec_add15_bottom 206
spec_add15_top 207
spec_add16_bottom 207
spec_add16_top 208
spec_add17_bottom 208
spec_add17_top 209
spec_add18_bottom 209
spec_add18_top 210

spec_add19_bottom 210
spec_add19_top 211
spec_add20_bottom 211
spec_add20_top 212
spec_add21_bottom 212
spec_add21_top 213
spec_add22_bottom 213
spec_add22_top 214
spec_add23_bottom 214
spec_add23_top 215
spec_add24_bottom 215
spec_add24_top 216
spec_add25_bottom 216
spec_add25_top 217
spec_add26_bottom 217
spec_add26_top 218
spec_add27_bottom 218
spec_add27_top 219
spec_add28_bottom 219
spec_add28_top 220
spec_add29_bottom 220
spec_add29_top 221
spec_add30_bottom 221
spec_add30_top 222
spec_add31_bottom 222
spec_add31_top 223
spec_add32_bottom 223
spec_add32_top 224
spec_add33_bottom 224
spec_add33_top 225
spec_add34_bottom 225
spec_add34_top 226
spec_add35_bottom 226
spec_add35_top 227
spec_add36_bottom 227
spec_add36_top 228
int_q1_status 228
int_q2_status 229
int_q3_status 230
int_q4_status 231
int_q5_status 232
int_q6_status 233
int_q7_status 234
int_q8_status 235
int_q9_status 236
int_q10_status 237
int_q11_status 238
int_q12_status 239
int_q13_status 240
int_q14_status 241
int_q15_status 242
transmit_q1_ptr 243
transmit_q2_ptr 244
transmit_q3_ptr 245
transmit_q4_ptr 246
transmit_q5_ptr 247
transmit_q6_ptr 248
transmit_q7_ptr 249
transmit_q8_ptr 250
transmit_q9_ptr 251

transmit_q10_ptr 252
transmit_q11_ptr 253
transmit_q12_ptr 254
transmit_q13_ptr 255
transmit_q14_ptr 256
transmit_q15_ptr 257
receive_q1_ptr 258
receive_q2_ptr 258
receive_q3_ptr 259
receive_q4_ptr 259
receive_q5_ptr 260
receive_q6_ptr 260
receive_q7_ptr 261
dma_rxbuf_size_q1 261
dma_rxbuf_size_q2 262
dma_rxbuf_size_q3 262
dma_rxbuf_size_q4 263
dma_rxbuf_size_q5 263
dma_rxbuf_size_q6 264
dma_rxbuf_size_q7 264
cbs_control 265
cbs_idleslope_q_a 265
cbs_idleslope_q_b 265
upper_tx_q_base_addr 266
tx_bd_control 266
rx_bd_control 266
upper_rx_q_base_addr 267
screening_type_1_register_0 267
screening_type_1_register_1 268
screening_type_1_register_2 268
screening_type_1_register_3 269
screening_type_1_register_4 269
screening_type_1_register_5 270
screening_type_1_register_6 270
screening_type_1_register_7 271
screening_type_1_register_8 271
screening_type_1_register_9 272
screening_type_1_register_10 272
screening_type_1_register_11 273
screening_type_1_register_12 273
screening_type_1_register_13 274
screening_type_1_register_14 274
screening_type_1_register_15 275
screening_type_2_register_0 276
screening_type_2_register_1 277
screening_type_2_register_2 278
screening_type_2_register_3 279
screening_type_2_register_4 280
screening_type_2_register_5 281
screening_type_2_register_6 282
screening_type_2_register_7 283
screening_type_2_register_8 284
screening_type_2_register_9 285
screening_type_2_register_10 286
screening_type_2_register_11 287
screening_type_2_register_12 288
screening_type_2_register_13 289
screening_type_2_register_14 290
screening_type_2_register_15 291
tx_sched_ctrl 292

bw_rate_limit_q0to3 294
bw_rate_limit_q4to7 294
bw_rate_limit_q8to11 295
bw_rate_limit_q12to15 295
tx_q_seg_alloc_q0to7 296
tx_q_seg_alloc_q8to15 297
receive_q8_ptr 298
receive_q9_ptr 298
receive_q10_ptr 299
receive_q11_ptr 299
receive_q12_ptr 300
receive_q13_ptr 300
receive_q14_ptr 301
receive_q15_ptr 301
dma_rxbuf_size_q8 302
dma_rxbuf_size_q9 302
dma_rxbuf_size_q10 303
dma_rxbuf_size_q11 303
dma_rxbuf_size_q12 304
dma_rxbuf_size_q13 304
dma_rxbuf_size_q14 305
dma_rxbuf_size_q15 305
int_q1_enable 306
int_q2_enable 307
int_q3_enable 308
int_q4_enable 309
int_q5_enable 310
int_q6_enable 311
int_q7_enable 312
int_q1_disable 313
int_q2_disable 314
int_q3_disable 315
int_q4_disable 316
int_q5_disable 317
int_q6_disable 318
int_q7_disable 319
int_q1_mask 320
int_q2_mask 321
int_q3_mask 322
int_q4_mask 323
int_q5_mask 324
int_q6_mask 325
int_q7_mask 326
int_q8_enable 327
int_q9_enable 328
int_q10_enable 329
int_q11_enable 330
int_q12_enable 331
int_q13_enable 332
int_q14_enable 333
int_q15_enable 334
int_q8_disable 335
int_q9_disable 336
int_q10_disable 337
int_q11_disable 338
int_q12_disable 339
int_q13_disable 340
int_q14_disable 341
int_q15_disable 342
int_q8_mask 343

int_q9_mask ... 344
int_q10_mask ... 345
int_q11_mask ... 346
int_q12_mask ... 347
int_q13_mask ... 348
int_q14_mask ... 349
int_q15_mask ... 350
screening_type_2_ethertype_reg_0 ... 351
screening_type_2_ethertype_reg_1 ... 351
screening_type_2_ethertype_reg_2 ... 351
screening_type_2_ethertype_reg_3 ... 351
screening_type_2_ethertype_reg_4 ... 352
screening_type_2_ethertype_reg_5 ... 352
screening_type_2_ethertype_reg_6 ... 352
screening_type_2_ethertype_reg_7 ... 352
type2_compare_0_word_0 ... 353
type2_compare_0_word_1 ... 354
type2_compare_1_word_0 ... 355
type2_compare_1_word_1 ... 356
type2_compare_2_word_0 ... 357
type2_compare_2_word_1 ... 358
type2_compare_3_word_0 ... 359
type2_compare_3_word_1 ... 360
type2_compare_4_word_0 ... 361
type2_compare_4_word_1 ... 362
type2_compare_5_word_0 ... 363
type2_compare_5_word_1 ... 364
type2_compare_6_word_0 ... 365
type2_compare_6_word_1 ... 366
type2_compare_7_word_0 ... 367
type2_compare_7_word_1 ... 368
type2_compare_8_word_0 ... 369
type2_compare_8_word_1 ... 370
type2_compare_9_word_0 ... 371
type2_compare_9_word_1 ... 372
type2_compare_10_word_0 ... 373
type2_compare_10_word_1 ... 374
type2_compare_11_word_0 ... 375
type2_compare_11_word_1 ... 376
type2_compare_12_word_0 ... 377
type2_compare_12_word_1 ... 378
type2_compare_13_word_0 ... 379
type2_compare_13_word_1 ... 380
type2_compare_14_word_0 ... 381
type2_compare_14_word_1 ... 382
type2_compare_15_word_0 ... 383
type2_compare_15_word_1 ... 384
type2_compare_16_word_0 ... 385
type2_compare_16_word_1 ... 386
type2_compare_17_word_0 ... 387
type2_compare_17_word_1 ... 388
type2_compare_18_word_0 ... 389
type2_compare_18_word_1 ... 390
type2_compare_19_word_0 ... 391
type2_compare_19_word_1 ... 392
type2_compare_20_word_0 ... 393
type2_compare_20_word_1 ... 394
type2_compare_21_word_0 ... 395
type2_compare_21_word_1 ... 396
type2_compare_22_word_0 ... 397

type2_compare_22_word_1 ... 398
type2_compare_23_word_0 ... 399
type2_compare_23_word_1 ... 400
type2_compare_24_word_0 ... 401
type2_compare_24_word_1 ... 402
type2_compare_25_word_0 ... 403
type2_compare_25_word_1 ... 404
type2_compare_26_word_0 ... 405
type2_compare_26_word_1 ... 406
type2_compare_27_word_0 ... 407
type2_compare_27_word_1 ... 408
type2_compare_28_word_0 ... 409
type2_compare_28_word_1 ... 410
type2_compare_29_word_0 ... 411
type2_compare_29_word_1 ... 412
type2_compare_30_word_0 ... 413
type2_compare_30_word_1 ... 414
type2_compare_31_word_0 ... 415
type2_compare_31_word_1 ... 416
Programming Examples ... 416
Initialization ... 416
Configuration ... 416
Receive Buffer List ... 416
Transmit Buffer List ... 417
Address Matching ... 418
PHY Maintenance ... 418
Interrupts ... 418
Transmitting Frames ... 419
Receiving Frames ... 419
Section 3 - Integrating the GEM_GXL ... 421
Introduction ... 421
IP Content ... 421
Reference Environment ... 422
Tools ... 422
Libraries ... 422
• Library tcbn28hpmbwp35hvtss0p81v125c ... 422
o 28nm slow: 0.81V, 125C ... 422
Implementation Application Notes ... 422
Parameterization ... 422
Common Configuration ... 424
Filter Configuration ... 425
Data path widths ... 426
DMA Configuration ... 427
TSU inclusion ... 430
Transmit and Receive Offload inclusion ... 431
Revision Register Values ... 431
Default Register Values ... 431
Other Configuration Parameters ... 432
Design Hierarchy ... 432
IP Application ... 433
APB Interface ... 433
AHB Master Interface ... 433
AXI Master Interface ... 433
MII/GMII ... 433
RGMII ... 433
In-band Status Decoding in RGMII Mode ... 433
RMII ... 434
MDIO ... 435
Other Interface Signals ... 435
Clock Requirements ... 435

Clocks 435
RMII Clocking 436
RGMII clocking 437
PCS Clocking 438
Resets 440
Metastability 440
Synthesis Constraints 441
Combinatorial Paths 441
Library Requirements 441
Verification & Test Application 441
Testbench structure 441
CPU Based Testing 442
Synthesizing the Design 442

Section 1 – Module Data

Scope

This user guide is provided to enable the GEM_GXL module to be programmed and fully integrated into an SoC design. It covers the following topics:

Module Data

Programming the IP

Integrating the IP

The GEM_GXL module also comes complete with the following:

Verilog HDL

Cadence RTL Compiler synthesis scripts

Verilog testbench

MAC Features

UNH tested

Compatible with IEEE Standard 802.3

10, 100 and 1000 Mbps operation

Statistics counter registers for RMON/MIB

RGMII, RMII, GMII, MII, TBI interfaces supported

Support for 10/100/1000/2500Mbps SGMII

Integrated 1000BASE-X physical coding sub-layer (PCS) with auto-negotiation and configurable PHY interfaces

Flexible Host Interface Selections Available

- External low latency FIFO interface
- AMBA AHB or AMBA AXI bus master direct memory access (DMA) or interface to external memory.
- AMBA AHB or AMBA AXI DMA with additional TX FIFO interface to inject fixed latency TX traffic.
- Configurable and/or programmable selection of host interface mode.

Support for up to sixteen priority queues on transmit and receive

Support for 802.1AS and 1588 precision clock synchronization protocol

Support for 1588 one step clock for TX Sync frames

Optional IEEE 1588 time stamp unit

Optional external time stamp interface port

Support for 802.3az Energy Efficient Ethernet

Full duplex flow control with recognition of incoming pause frames and hardware generation of transmitted pause frames

Support for 802.1Qbb priority-based flow control

Half duplex flow control by forcing collisions on incoming frames

Receive and transmit IP, TCP and UDP checksum offload. Both IPv4 (with IP options) and IPv6 (with extension headers) packet types supported

Automatic pad and cyclic redundancy check (CRC) generation on transmitted frames

Address checking logic for up to thirty six specific 48 bit SA or DA addresses with byte masks, four type Ids, promiscuous mode, external address checking, hash matching of unicast and multicast destination addresses and Wake-on-LAN

MDIO interface for physical layer management

Support for jumbo frames

Programmable IPG stretch

Configuration interface compatible with AMBA APB interface, version 2.0

Interrupt generation to signal receive and transmit completion, errors and other events

Support for 802.1Q VLAN tagging with recognition of incoming VLAN and priority tagged frames

Supports 802.1Qav traffic shaping on two highest priority transmit queues

Supports a choice of strict priority, Deficit Weighted Round Robin (DWRR) or Enhanced Transmission Selection (ETS - 802.1Qaz) on all transmit queues

Optional DMA Features

AHB or high performance AXI4

High performance AXI4 features

- Support for very high latency systems.
    - Support for multiple outstanding transactions on read address and write address channels.
    - Outstanding transactions can span multiple frames.
- Programmable burst length up to 256 beats.
- Automatic burst breaks at 4KB boundaries.
- Options to help maximize burst efficiency.

AHB features

- Programmable burst length up to 16 beats.
- Automatic burst breaks at 1KB boundaries.
- Options to help maximize burst efficiency.

Selectable low latency FIFOs or packet buffer SRAM on transmit and receive DMA

Support for full store and forward or flexible low latency partial store and forward configurations available

Configurable for 32 bit, 64 bit and 128 bit data paths (128 bit AHB is not supported)

Configurable for 32 bit and 64 bit address bus width

Various optional advanced TCP/IP layer offload features to reduce CPU frame processing overhead.

- Large Send Offload (LSO)
    - TCP Segmentation Offload (TSO).
    - UDP fragmentation Offload (UFO)
- Receive Side Coalescing (RSC).
- Receive Header / Data Splitting.
- Interrupt moderation

Support for Priority Queue Management to stream data to and from up to 16 independent external traffic queues

- Highly flexible and configurable screening algorithm based on VLAN, IP, TCP, UDP and other indexed packet fields to map receive frames to priority queues.
- High priority traffic given precedence over lower priority traffic on TX path.
- Ability for high priority transmit traffic to be internally reordered to reduce traffic latency.

Support for either dual port SRAM or single port SRAM.

Programmable endianism

Automatic self completing retries for half duplex operation

Optional automatic discard of frames received with MAC related errors

Optional support for manual or automatic flushing of frames received when external memory resource has been exhausted or AHB/AXI access errors occur

Extended Buffer Descriptor Modes:

- Extended Buffer Addresses in 64bit addressing modes require larger descriptors.
- Support for capturing timestamp of tx or rx frame within Buffer Descriptor
- Ability to configure type of frame for capture timestamp (ptp event or ptp general or all frames or none)

Description

The Gigabit Ethernet MAC (GEM_GXL) module implements a 10/100/1000 Mbps Ethernet MAC compatible with the IEEE 802.3 standard. The GEM_GXL can operate in either half or full duplex. The network configuration register is used to select the speed, duplex mode and interface type (MII, GMII, RGMII, TBI or SGMII).

The GEM_GXL comprises the following constituent components:

MAC controlling transmit, receive, address checking and loopback

REG_TOP providing control and status registers, statistics registers and synchronization logic

PCS controlling 8B/10B encode/decode, PCS transmit, PCS receive and PCS auto-negotiation

DMA_TOP controlling DMA transmit, DMA receive and AHB/AXI arbitration logic

TSU calculates the IEEE 1588 timer values

In system applications where no DMA operation is required, the DMA_TOP module can be removed using a compile option and an external FIFO interface used to incorporate the GEM into a SoC environment. The GEM can also be configured to include the DMA in addition to a Low Latency FIFO interface. The GEM_PCS module may also be removed using a compile option.

MAC

The MAC transmit block takes data from the external FIFO interface, adds preamble and, if necessary, pad and frame check sequence (FCS). Both half duplex and full duplex Ethernet modes of operation are supported. When operating in half duplex mode, the MAC transmit block generates data according to the carrier sense multiple access with collision detect (CSMA/CD) protocol. The start of transmission is deferred if carrier sense (`crs`) is active. If collision (`col`) becomes active during transmission, a jam sequence is asserted and the transmission is re-tried after a random back off. The `crs` and `col` signals have no effect in full duplex mode.

The MAC receive block checks for valid preamble, FCS, alignment and length, and presents received frames to the MAC address checking block and external FIFO interface. Software can configure the GEM_GXL to receive jumbo frames up to 16,383 bytes. It can optionally strip CRC from the received frame prior to transfer to the external FIFO interface.

The address checker recognizes a configurable number of maskable source or destination specific 48 bit addresses, can recognize four different type ID values, and contains a 64 bit hash register for matching multicast and unicast addresses as required. It can recognize the broadcast address of all ones, copy all frames and act on external address matching signals. The MAC can also reject all frames that are not VLAN tagged and recognise Wake on LAN events. Address comparison against individual bits of Specific Address Register 1 can be masked by means of the Specific Address Mask Register. All other specific address filters are byte maskable.

The MAC receive block supports offloading of IP, TCP and UDP checksum calculations (both IPv4 and IPv6 packet types supported), and can automatically discard bad checksum frames.

The MAC can optionally replace the timestamp field in PTP 1588 transmit sync frames to support one step clock mode.

The MAC can optionally do one step transparent clock residence time correction for PTP 1588 version 2 transmit sync frames

PCS

A PCS sub-layer is incorporated for 1000BASE-X operation and provides a configurable interface to the PMA layer. A PCS transmit module and 8B/10B encoder form the transmit path, and a PCS receiver and two 8B/10B decoders form the receive path, which is expanded into 16 bits to simplify clocking. Auto-negotiation is also provided for network configuration. The PMA interface may be configured to be 10 or 20-bits and appropriate gearboxes for datawidth conversion will be automatically instantiated.

The PCS can be configured through the network configuration register to support an SGMII interface between the MAC and an external PHY. For a complete SGMII solution a SerDes hard macro is also required.

REG_TOP

Control registers drive the management data input/output (MDIO) interface, set-up DMA activity, start frame transmission and select modes of operation such as full duplex, half duplex and 10/100/1000 Mbps operation. The register interface is compatible with the AMBA APB bus standard. Where possible, register set compatibility has been maintained with the Ethernet MAC 10/100 (MACB) from Cadence.

The statistics register block contains registers for counting various types of event associated with transmit and receive operation. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3 clause 30.

DMA_TOP

The DMA may be configured in either a very low latency buffering mode using internal transmit and receive FIFOs or a packet buffering mode using external memories.

Data path bus widths of 32 bit, 64 bit and 128 bit are supported at all data rates (128 bit AHB is not supported) although system designers should carefully consider the bandwidth requirements of the external data path being used — either external FIFO interface or AMBA AHB/AXI interface . The DMA block connects to external memory through its AMBA AHB or AXI bus interface. The DMA loads the transmit buffer and empties the receive buffer using bus master operations. Receive data is not sent to memory until the address checking logic has determined that the frame should be copied.

Receive or transmit frames are stored in one or more DMA buffers. The receive buffer size is programmable between 64 bytes and 16 Kbytes. Transmit buffers range in length between 1 and 16383 bytes, and up to 128 buffers are permitted per frame. The DMA block manages the transmit and receive frame buffer queues.

For chip architecture where DMA is not required, the GEM is supplied with an external transmit and receive external FIFO interface, thus simplifying design complexity and reducing gate count.

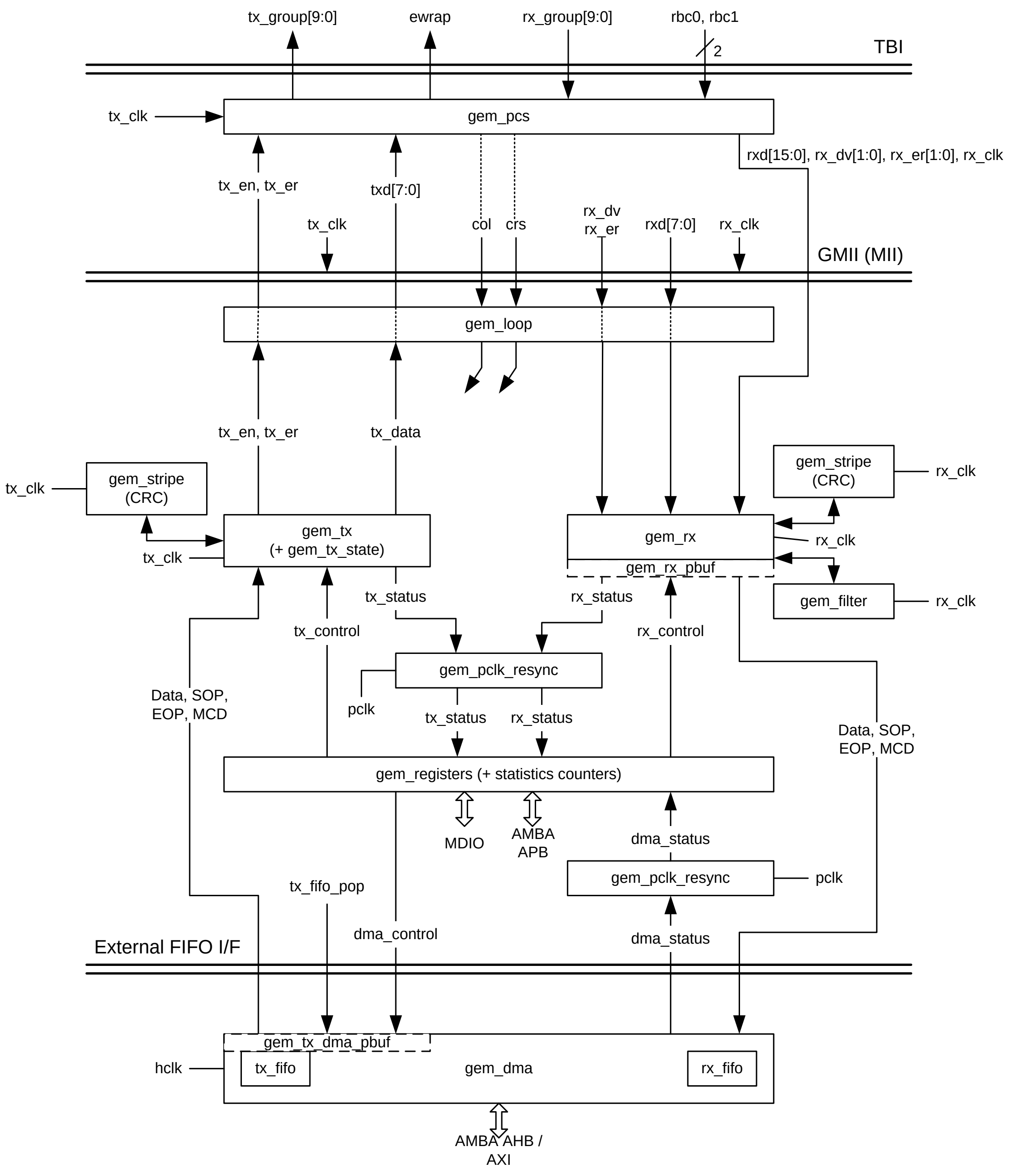
Note the DMA block does not use AMBA AHB split and retry operations.

TSU

The 1588 time stamp unit (TSU) is a timer implemented as a 102 bit register. The 48 upper bits count seconds and the next 30 lower bits count nanoseconds and the lowest 24 bits count sub nanoseconds. The 54 lower bits roll over when they have counted to one second. The timer increments by a programmable period (to approximately 58.6 attosecond resolution – 5.86E-17) with each PCLK period and can also be adjusted in 1ns resolution (incremented or decremented) through APB register accesses.

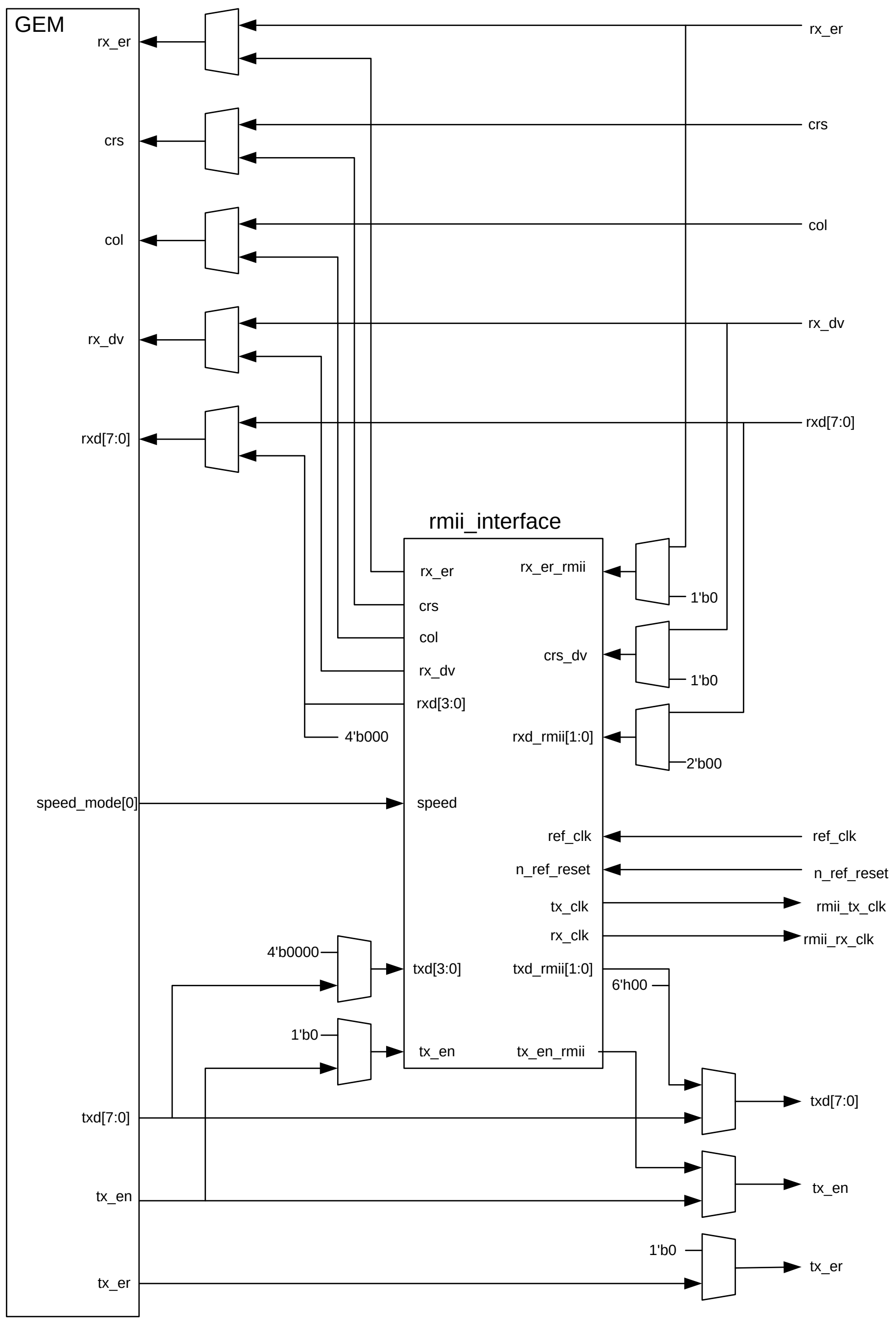
An optional external TSU port can be built into the design using a `define. When the external TSU port is available, the selection between internal or external TSU is performed using a programmable register. The default selection is internal TSU. The external port is 94 bits wide, conforming to the most significant bits of the internal TSU count as defined above.

Block Diagram

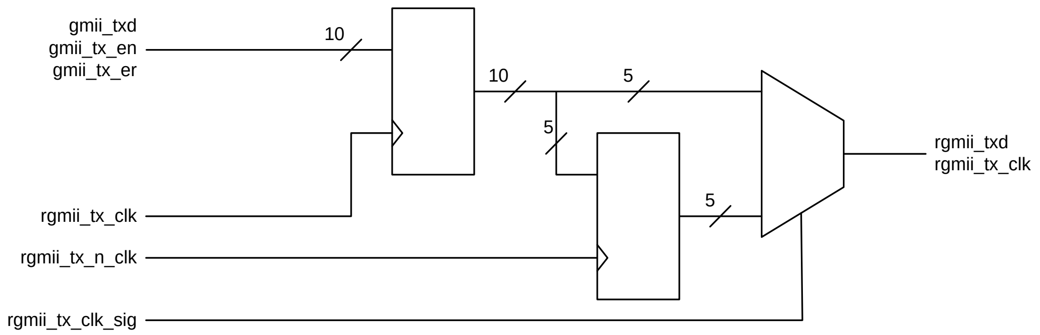


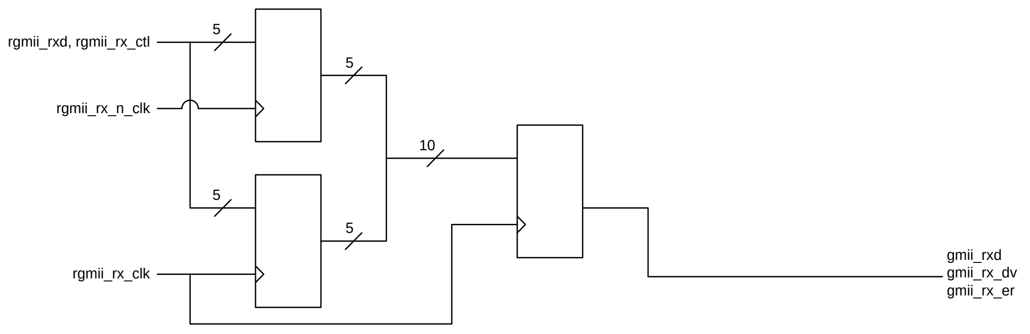
The block diagrams above are true of a GMII interface. The GEM is also capable of supporting RMII and RGMII interfaces.

When the RMII interface is included (selected as a configuration option, refer to the configuration section of this document for details), it is added to the GEM as follows:

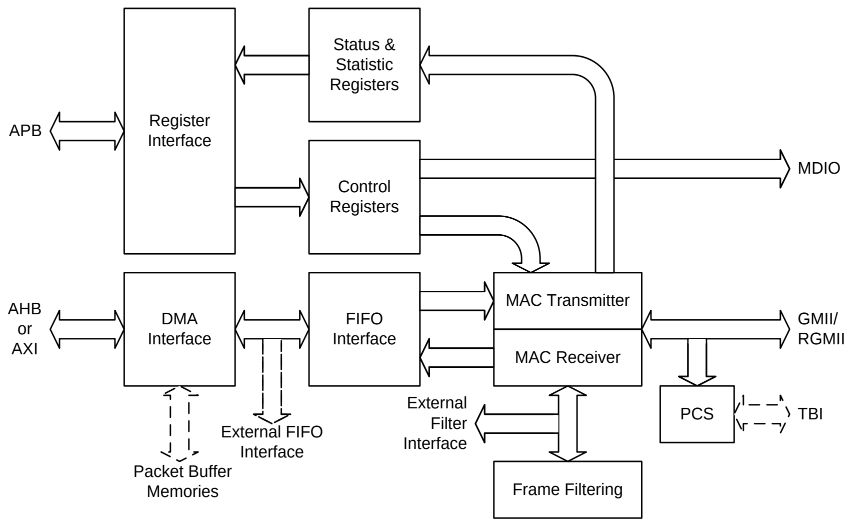


When the RGMII interface is configured (refer to the configuration section for details), it is implemented as a bridge function (GMII to RGMII) and replaces the standard GMII interface.





Signal Interfaces



The GEM_GXL includes the following signal interfaces:

- GMII, RGMII, MII, RMII , and TBI to an external PHY
- MDIO interface for external PHY management
- AMBA Advanced Peripheral Bus (APB) slave interface for accessing the GEM_GXL's registers
- AMBA Advanced High Speed Bus (AHB or AXI4) master interface for memory access
- An optional FIFO interface in applications where DMA functionality is not required
- An optional packet buffer memory interface for the DMA
- An optional timestamp interface

The AMBA interfaces fully conform to the ARM AMBA Revision 2.0 specification.

Ethernet Interface (GMII/MII)

Signal Name	I/O	Description
`col`	I	Collision detect from the PHY (asynchronous).
`crs`	I	Carrier sense from the PHY (asynchronous).
`tx_er`	O	Transmit error signal to the PHY. Asserted if the DMA block fails to fetch data from memory during frame transmission (tx_clk timed).
`txd[7:0]`	O	Transmit data to the PHY. 10/100 mode: txd[3:0] used, txd[7:4] tied to logic 0. 1000 mode: txd[7:0] used (tx_clk timed).
`tx_en`	O	Transmit enable to the PHY (tx_clk timed).
`tx_clk`	I	Transmit clock from the system clock generator. 312.5MHz for 2.Gbps operation 125MHz for gigabit operation 25MHz for 100M operation 2.5MHz for 10M operation 12.5MHz for 100M SGMII operation 1.25MHz for 10M SGMII operation
`rxd[7:0]`	I	Receive data from the PHY: 10/100 mode: rxd[3:0] used, rxd[7:4] not used. 1000 mode: rxd[7:0] used.
`rx_er`	I	Receive error signal from the PHY.
`rx_clk`	I	Receive clock from the system clock generator. 125MHz for gigabit GMII operation 25MHz for 100M operation 2.5MHz for 10M operation 62.5MHz for gigabit SGMII/TBI operation 12.5MHz for 100M SGMII operation 1.25MHz for 10M SGMII operation
`rx_dv`	I	Receive data valid signal from the PHY.

Ethernet Interface (RGMII)

Signal Name	I/O	Description
`rgmii_tx_clk_sig`	I	Slightly delayed version of `tx_clk` used as the signal to control the multiplexer for `rgmii_txd` output data. Can be set high for 10/100M operation to prevent rgmii_txd driving zero on the falling edge of tx_clk.
`rgmii_txd[3:0]`	O	Transmit data signal to the PHY.
`rgmii_tx_ctl`	O	Transmit control signal to the PHY.
`rgmii_rxd[3:0]`	I	Receive data signal from the PHY.
`rgmii_rx_ctl`	I	Receive control signal from the PHY.
`tx_clk`	I	Transmit clock from the system clock generator. 125MHz for gigabit GMII operation 25MHz for 100M operation 2.5MHz for 10M operation This clock must also be output from the SoC to support source synchronous clocking to the PHY.

`n_tx_clk`	I	Inverted tx_clk
`rx_clk`	I	Receive clock from the PHY. 125MHz for gigabit GMII operation 25MHz for 100M operation 2.5MHz for 10M operation
`n_rx_clk`	I	Inverted rx_clk
`rgmii_speed[1:0]`	O	RGMII extracted signal indicating speed of operation. This output is generated synchronous to `rx_clk`.
`rgmii_link_status`	O	RGMII extracted link status signal. This output is generated synchronous to `rx_clk`.
`rgmii_duplex_out`	O	RGMII extracted signal indicating duplex mode. This output is generated synchronous to `rx_clk`.

Ethernet Interface (RMII)

Signal Name	I/O	Description
`mii_select`	I	Drive high to select standard GMII interface, also drive high in gigabit mode. Drive low for RMII operation.
`ref_clk`	I	50 MHz reference clock for RMII interface module.
`n_ref_reset`	I	Active low asynchronous reset for the `ref_clk` domain
`rmii_tx_clk`	O	`ref_clk` timed signal to be used as a source for `tx_clk`.
`rmii_rx_clk`	O	`ref_clk` timed signal to be used as a source for `rx_clk`.
`rmii_crs_dv`	I	Receive data valid signal from the PHY
`rmii_txd[7:0]`	O	Transmit data to the PHY.
`rmii_tx_en`	O	Transmit enable to the PHY (tx_clk timed).
`rmii_rxd[1:0]`	I	Receive data from the PHY.
`rmii_rx_er`	I	Receive error signal from the PHY.

MDIO Interface

Signal Name	I/O	Description
`mdc`	O	Management data clock to pin. (pclk timed)
`mdio_in`	I	Management data input from MDIO pin.
`mdio_out`	O	Management data output to MDIO pin. (pclk timed)

`mdio_en`	O	Management data output enable to MDIO pin (pclk timed). At the top-level the three mdio output pins can be used to drive a single tri-state pin. Alternatively MDIO may be implemented as a pull-down. In this case the enable to the pull-down should be driven with ~mdio_out & mdio_en. Example top-level RTL:
		`assign mdio_in = mdio;`
		`assign mdio = (mdio_en) ? mdio_out : 1'bz;`
		or if an external pull-up is used:
		`assign mdio = (mdio_en & ~mdio_out) ? 1'b0 : 1'bz;`

Control/Status Interface

Signal Name	I/O	Description
`loopback`	O	Selects external loopback — not used by most PHYs. This signal does not need to be bonded out, but is used by the testbench.
`loopback_local`	O	Internal loopback indication to the system clock generator.
`half_duplex`	O	Selects half duplex — this signal does not need to be bonded out, but is used by the testbench. It is also used by the RGMII core, half_duplex is inverted and connected to gmii_duplex_en of the RGMII core.
`speed_mode[3:0]`	O	Indicates speed and external interface that the GEM is currently configured to use to the system clock generator: 0000 - 10 Mbps Ethernet operation using MII interface 0001 - 100 Mbps operation using MII interface 001x - 1000 Mbps operation using GMII interface 0100 - 10 Mbps operation using SGMII interface 0101 - 100 Mbps operation using SGMII interface 011x - 1000 Mbps operation using TBI or SGMII interface speed_mode[2:0] indicates the value of the tbi, gigabit and speed bits 11, 10 and 0 of the network configuration register.
`tx_pause`	I	Toggle input for pause frame transmission. When `tx_pfc_sel` is LOW, a classic 802.3 pause frame will be transmitted. The pause quantum value to be transmitted depends on the state of the `tx_pause_zero pin.` When `tx_pfc_sel` is HIGH, an 802.1Qbb, or PFC priority based pause frame will be transmitted. The pause quantum value to be transmitted depends on the state of the tx_pfc_priority bus. Tie this input low if hardware initiated flow control is not being used. This asynchronous input is synchronized into the tx_clk domain.
`tx_pfc_sel`	I	Indicates whether the pause frame to be transmitted should be classic 802.3 (when set to logic 0) or PFC priority based pause frame (when set to logic 1). This signal should be valid when tx_pause toggles and remain valid for at least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`tx_pause_zero`	I	Indicates whether the classic 802.3 pause frame to be transmitted is to have zero pause quantum (when set to logic 1) or the value of the transmit pause quantum register (when set to logic 0). This signal should be valid when tx_pause toggles and remain valid for at

		least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`tx_pfc_pause[7:0]`	I	Indicates the value to be used in the 8bit priority enable vector of the PFC priority based pause frame. This signal should be valid when tx_pause toggles and remain valid for at least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`tx_pfc_pause_zero [7:0]`	I	Indicates whether each priority entry in the PFC priority based pause frame is to have zero pause quantum (when set to logic 1) or the value of the transmit pause quantum register (when set to logic 0). This signal should be valid when tx_pause toggles and remain valid for at least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`trigger_dma_tx_start`	I	Toggle input for starting transmit DMA. Only present when using the DMA configured for internal packet buffering. This signal is used as a hardware alternative to setting the TX START bit of the network control register. If the user only wishes to use software to initiate transmission, then this input can be tied low. This asynchronous input is synchronized into the hclk domain.
`rx_pfc_paused[7:0]`	O	Each bit corresponds to a priority indicated within the PFC priority based pause frame. Each bit is set when a PFC priority based pause frame has been received, and the associated priority pause time quantum is non-zero. Each bit is cleared when the associated pause time identified by the received pause time quantum has elapsed.
`pfc_negotiate`	O	Identifies that PFC priority based pause flow control has been negotiated. 0 - No PFC priority based pause frames have yet been received, flow control is being handled using classic 802.3 pause frames. 1 – At least one PFC priority based pause frames has been received. All subsequent 802.3 pause frames will be dropped.
`rx_databuf_wr_q0`	O	When priority queuing is disabled, this output is toggled on a DMA write to the first word of each DMA data buffer. When priority queuing is enabled, this output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 0. This output can be left unconnected.
`rx_databuf_wr_q1`	O	Only present when priority queuing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 1.

rx_databuf_wr_q2	O	Only present when priority queuing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 2.
rx_databuf_wr_q3	O	Only present when priority queuing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 3.
rx_databuf_wr_q4	O	Only present when priority queuing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 4.
rx_databuf_wr_q5	O	Only present when priority queuing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 5.
rx_databuf_wr_q6	O	Only present when priority queuing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 6.
rx_databuf_wr_q7	O	Only present when priority queuing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 7.
halfduplex_flow_c ontrol_en	I	Asynchronous Input. This input is ignored when the GEM is configured for full duplex or gigabit modes. Setting this bit high will enable the half duplex flow control mechanism. The transmit block will transmit 64 bits of data, whenever it sees an incoming frame in order to force a collision. Tie this signal low if not being used
dma_bus_width[1:0]	O	Indicates width of DMA or external FIFO data bus as follows— this signal does not need to be bonded out, but is used by the testbench: 00 - 32 bit data bus 01 - 64 bit data bus 10 or 11 - 128 bit data bus.

External Filter Interface

Signal Name	I/O	Description
ext_match1 ext_match2 ext_match3 ext_match4	I	Optional external match 1, 2, 3 and 4 — any one of these input signals from the external filter logic can be asserted indicating a match was found. Filter matches can take place on either ext_sa, ext_da, ext_type or ext_vid. If bit 26 ext_rxq_sel_en is set in the network control register then these inputs select which queue a matched receive frame is sent to. Synchronous to rx_clk.
ext_sa[47:0]	O	Source address for comparison, extracted from packet being received.
ext_sa_stb	O	Validates source address.

`ext_da[47:0]`	O	Destination address for comparison, extracted from packet being received.
`ext_da_stb`	O	Validates destination address.
`ext_type[15:0]`	O	Length/type field for comparison, extracted from packet being received.
`ext_type_stb`	O	Validates length/type field.
`ext_vlan_tag1[31:0]`	O	VLAN identifier and VLAN ID field for comparison, extracted from packet being received. For packets incorporating 2 VLAN tags (stacked VLAN), this represents the first VLAN tag
`ext_vlan_tag1_stb`	O	Validates VLAN ID field.
`ext_vlan_tag2[31:0]`	O	VLAN identifier and VLAN ID field for comparison, extracted from packet being received. For packets incorporating 2 VLAN tags (stacked VLAN), this represents the second VLAN tag. These bits are only valid for packets incorporating the stacked VLAN processing feature.
`ext_vlan_tag2_stb`	O	Validates VLAN ID field.
`ext_ip_sa[127:0]`	O	IP source address for comparison, extracted from the packet being received. For IPv4 packets, only bits [31:0] are valid. For Ipv6 packets, bits [127:0] are valid.
`ext_ip_sa_stb`	O	Validates IP source address field.
`ext_ip_da[127:0]`	O	IP destination address for comparison, extracted from the packet being received. For IPv4 packets, only bits [31:0] are valid. For Ipv6 packets, bits [127:0] are valid.
`ext_ip_da_stb`	O	Validates IP destination address field.
`ext_source_port[15:0]`	O	TCP/UDP source port number for comparison.
`ext_sp_stb`	O	Validates ext_source_port.
`ext_dest_port[15:0]`	O	TCP/UDP destination port number for comparison.
`ext_dp_stb`	O	Validates ext_dest_port.
`ext_ipv6`	O	Asserted high for ipv6 frames
`wol`	O	Asserted for 64 `rx_clk` cycles if a magic packet, ARP request, specific address 1 or multicast hash event is decoded and is enabled through the Wake on LAN register. Only bond this pin out if Wake on LAN functionality is required.

IEEE 1588 PTP frame recognition and Time Stamp Unit

Signal Name	I/O	Description

`sof_tx`	O	Asserted high synchronous to tx_clk when the SFD is detected on a transmit frame, deasserted at end of frame
`sync_frame_tx`	O	Asserted high synchronous to tx_clk if PTP sync frame is detected on transmit.
`delay_req_tx`	O	Asserted high synchronous to tx_clk if PTP delay request frame is detected on transmit.
`pdelay_req_tx`	O	Asserted high synchronous to tx_clk if PTP peer delay request frame is detected on transmit.
`pdelay_resp_tx`	O	Asserted high synchronous to tx_clk if PTP peer delay response frame is detected on transmit.
`sof_rx`	O	Asserted high synchronous to rx_clk when the SFD is detected on a receive frame
`sync_frame_rx`	O	Asserted high synchronous to rx_clk if PTP sync frame is detected on receive.
`delay_req_rx`	O	Asserted high synchronous to rx_clk if PTP delay request frame is detected on receive.
`pdelay_req_rx`	O	Asserted high synchronous to rx_clk if PTP peer delay request frame is detected on receive.
`pdelay_resp_rx`	O	Asserted high synchronous to rx_clk if PTP peer delay response frame is detected on receive.
`tsu_clk`	I	Alternative clock source for the time stamp unit. If gem_tsu_clk is defined in the gem_gxl_defs.v file then the TSU is clocked by tsu_clk rather than pclk. This clock must have a frequency greater than 1/8th the frequency of tx_clk or rx_clk. Timestamp accuracy improves with higher frequencies.
`gem_tsu_ms`	I	TSU master/slave. Used with gem_tsu_inc_ctrl to control incrementing of the TSU and loading the sync strobe register.
`gem_tsu_inc_ctrl[1:0]`	I	Used to control incrementing of the TSU and synchronous to tsu_clk or pclk. Drive high when not being used.
`tsu_timer_cnt[93:0]`	O	TSU timer count value, synchronized to tsu_clk or pclk. Upper 48 bits are seconds value and lower 46 bits are nanoseconds / sub-nanoseconds. Bit 46 toggles every second, i.e. 1 pps (also bit 45's falling edge is co-incident with this toggle).
`tsu_timer_cmp_val`	O	TSU timer comparison valid, synchronized to tsu_clk or pclk. Asserted high when upper 70 bits of TSU timer count value are equal to programmed comparison value.
`ext_tsu_timer[93:0]`	I	A 94 bit external timestamp interface enabled using the Network_Control Register. This port must be synchronous to the tsu_clk

Ethernet Interface (TBI)

Signal Name	I/O	Description
`gtx_clk`	I	PCS transmit clock running at 125MHz for Gigabit operation. This clock uses the same clock source as `tx_clk`. When operating in SGMII applications `tx_clk` is derived from this clock and divided down appropriately for 10 and 100Mb/s speeds
`gtx20_clk`	I	Optional interface clock used when the `gem_pcs_20b_if` define is used to select a 20-bit PHY interface. This clock is half the frequency of `gtx_clk` and both clocks must be phase aligned.
`pcs_rx_clk`	I	Optional interface clock for when either gem_pcs_10b_if or gem_pcs_20b_if is defined. This clock is 62.5MHz for Gigabit operation. When operating in SGMII applications rx_clk is derived from either this clock or rbc1 and divided down appropriately for 10 and 100Mb/s speeds
`pcs_rx10_clk`	I	Optional interface clock used when the gem_pcs_10b_if define is used to select a 10-bit PHY interface. This clock is twice the frequency of pcs_rx_clk and both clocks must be phase aligned.
`rbc0` `rbc1`	I	These optional clocks are used if gem_pcs_legacy_if is defined to select a TBI interface as defined in IEEE 802.3 Clause 36. The clocks will be 62.5MHz and `rbc1` will be 180 degrees phase shifted with respect to `rbc0`.
`tx_group[n:0]`	O	8B/10B encoded transmit data to PHY transceiver, synchronised to either gtx_clk or gtx20_clk depending on interface configuration. The interface width will either be 10 or 20-bits.
`rx_group[n:0]`	I	8B/10B encoded receive data from PHY transceiver. The interface width will either be 10 or 20-bits and the data should be synchronous to rbc0/rbc1 clocks if gem_pcs_legacy_if is defined otherwise depending on interface width will be synchronous to either pcs_rx_clk or pcs_rx10_clk.
`pcs_cal_bypas` `s`	I	Optional static control to enable bypass of comma alignment module when gem_pcs_legacy_if is not used. Unless an external comma alignment function is used, this signal should always be tied to 1'b0.
`pcs_cgalign_b` `ypass`	I	Optional static control to enable bypass of codegroup alignment module when gem_pcs_legacy_if is not used. If a 10-bit interface is used, it is recommended that this signal is always tied to 1'b0.
`ewrap`	O	Control for loopback operation in the PHY transceiver.
`en_cdet`	O	Control for comma alignment in the PHY transceiver (see IEEE 802.3 subclause 36.3.2.4). Leave unconnected if an on-chip SerDes is being used.

`signal_detect`	I	Valid signal detected from PMA. This signal must be driven high for PCS synchronization to occur. Users are advised to check that this signal is driven high by their SerDes or to provide a control signal to over-ride the signal from the SerDes.

Interrupt Controller Interface

Signal Name	I/O	Description
`ethernet_int`	O	Ethernet interrupt signal synchronous to `pclk`. When priority queuing is enabled, this interrupt represents the interrupt dedicated to Q0. When priority queuing is disabled, this interrupt represents the only interrupt output.
`ethernet_int_q*`	O	Ethernet interrupt signal synchronous to `pclk`. There is one interrupt port per priority queue (excluding queue 0).
`ext_interrupt_in`	I	External interrupt input pin, the core can be programmed to generate an interrupt on the rising edge of this pin. Tie this signal low if not being used. This asynchronous input is synchronized into the pclk domain.

External Transmit FIFO Interface

Signal Name	I/O	Description
`tx_r_data_rdy`	I	When set to logic 1, indicates enough data is present in the external FIFO for Ethernet frame transmission to commence on the current packet.
`tx_r_rd`	O	Single `tx_clk` clock cycle wide active high output requesting a word (either 32-bits, 64-bits or 128-bits) of information from the external FIFO interface. Synchronous to the `tx_clk` clock domain.
`tx_r_valid`	I	Single `tx_clk` clock cycle wide active high input indicating requested FIFO data is now valid. Validates the following inputs: `tx_r_data[127:0]`, `tx_r_sop`, `tx_r_eop`, `tx_r_err` and `tx_r_mod[3:0]`.
`tx_r_data[127:0]`	I	FIFO data for transmission, either 32-bit, 64-bit or 128-bit depending on `dma_bus_width[2:0]` which is configured through the register interface. This input is only valid whilst `tx_r_valid` is high.
`tx_r_sop`	I	Start of packet, indicates the word received from the external FIFO interface is the first in a packet. This input is only valid whilst `tx_r_valid` is high.

tx_r_eop	I	End of packet, indicates the word received from the external FIFO interface is the last in a packet. This input is only valid whilst tx_r_valid is high.
tx_r_mod[3:0]	I	Read module, indicates how many bytes are valid on tx_r_data during the last transfer of the packet: 0000- all bytes are valid 0001 - tx_r_data[7:0] is valid 0010 - tx_r_data[15:0] is valid 0011 - tx_r_data[23:0] is valid, and so on until 1111 - tx_r_data[119:0] is valid. This input is only valid when both tx_r_eop and tx_r_valid are high.
tx_r_err	I	Error, active high input indicating the current packet contains an error. This signal is only valid whilst tx_r_valid is high and may be set at any time during the packet transfer. If tx_r_err is set concurrently with tx_r_sop then no frame will be transmitted and dma_tx_end_tog will not be toggled.
tx_r_underflow	I	FIFO under flow, indicating the transmit FIFO was empty when a read was attempted. This signal is only valid when the GEM has attempted a read by asserting tx_r_rd and the tx_r_valid signal has not yet been indicated. tx_r_flushed should be asserted following this event to indicate to the GEM when it is safe to resume reading.
tx_r_flushed	I	This signal must be driven high and then low after a major error event to indicate to GEM that the external FIFO has been flushed. This will enable the GEM to resume reading data. Events that require this to be set are indicated by any bit of tx_r_status being asserted
tx_r_control	I	tx_no_crc, set active high at start of packet to indicate current frame is to be transmitted without crc being appended. This input is only valid whilst both tx_r_valid and tx_r_sop are high.

Note: *The transmit FIFO interface inputs must be pre-synchronized to the tx_clk clock domain.*

External Transmit FIFO Interface Status

Signal Name	I/O	Description
dma_tx_end_tog	O	Toggled to indicate that a frame has been completed and status is now valid on the tx_r_status and tx_r_timestamp outputs. Note that this signal is not activated when a frame is being retried due to a collision.
dma_tx_status_tog	I	This signal must be toggled each time either dma_tx_end_tog or collision_occured are activated, to indicate that the status has been acknowledged. This asynchronous input is synchronized into the tx_clk domain.
tx_r_timestamp[77:0]	O	Timestamp value at sop for transmit frame 48b seconds, 30b nanoseconds.

tx_r_status[3:0]	O	[3]: fifo_underrun - status output indicating that the MAC transmitter has under run due to one of the following conditions. Data under run indicated by tx_r_underflow input from the external FIFO interface, status write back error due to status write back not completing when another status write back is attempted or a tx_r_err was driven on the external FIFO interface during the last frame transfer. Reset once dma_tx_status_tog changes logic state. [2]: collision_occured – HALF DUPLEX status output indicating that the frame in progress has suffered a collision and that re-transmission of the frame should take place. Set when the collision occurs and reset once dma_tx_status_tog changes logic state. Note that dma_tx_end_tog is not activated. (Note: This bit is not driven when the external TX FIFO interface is used in conjunction with the DMA (when `gem_tx_add_fifo_if verilog define is set) [1]: late_coll_occured – HALF DUPLEX status output indicating that the frame in progress suffered a late collision and was aborted. Valid when dma_tx_end_tog changes logic state. Cleared once dma_tx_status_tog changes logic state. . . (Note: This bit is not driven when the external TX FIFO interface is used in conjunction with the DMA (when `gem_tx_add_fifo_if verilog define is set) [0]: too_many_retries – HALF DUPLEX status output indicating that the frame in progress experienced excess collisions and was aborted. Valid when dma_tx_end_tog changes logic state. Cleared once dma_tx_status_tog changes logic state. . . (Note: This bit is not driven when the external TX FIFO interface is used in conjunction with the DMA (when `gem_tx_add_fifo_if verilog define is set)

External Receive FIFO Interface

Note: *Receive FIFO interface outputs are synchronized to the rx_clk clock domain.*

Signal Name	I/O	Description
rx_w_wr	O	Single rx_clk clock cycle wide active high output indicating a write to the external FIFO interface.
rx_w_data[127:0]	O	Received data for output to the external FIFO interface, either 32-bit, 64-bit or 128-bit depending on dma_bus_width[2:0] which is configured through the register interface. This output is only valid when rx_w_wr is high.
rx_w_sop	O	Start of packet, indicates the word output to the external FIFO interface is the first in a packet. This output is only valid when rx_w_wr is high.

rx_w_eop	O	End of packet, indicates the word output to the external FIFO interface is the last in a packet. This output is only valid when rx_w_wr is high. (Note this signal will not be asserted if the receive path is disabled during frame reception however rx_w_flush will be asserted.)
rx_w_status[44:0]	O	Status signals, valid with rx_w_eop (with the exception of bits 20:15, which are valid on both rx_w_sop and rx_w_eop), definitions as follows: [44] - rx_w_code_error indicates a code error. [43] - rx_w_too_long indicates the frame was too long. [42] - rx_w_too_short indicates the frame was too short. [41] - rx_w_crc_error indicates the frame had a bad crc. [40] - rx_w_length_error indicates the length field was checked and was incorrect. [39] - rx_w_snap_match indicates the frame was SNAP encoded and had either no VLAN tag or a VLAN tag with the CFI bit not set. [38] - rx_w_checksumu indicates the UDP checksum was checked and was correct. [37] - rx_w_checksumt indicates the TCP checksum was checked and was correct. [36] - rx_w_checksumi indicates the IP checksum was checked and was correct. [35] - rx_w_type_match4, indicates the received frame was matched on type ID register 4. [34] - rx_w_type_match3, indicates the received frame was matched on type ID register 3. [33] - rx_w_type_match2, indicates the received frame was matched on type ID register 2. [32] - rx_w_type_match1, indicates the received frame was matched on type ID register 1. [31] - rx_w_add_match4, indicates the received frame was matched on specific address register 4. [30] - rx_w_add_match3, indicates the received frame was matched on specific address register 3. [29] - rx_w_add_match2, indicates the received frame was matched on specific address register 2. [28] - rx_w_add_match1, indicates the received frame was matched on specific address register 1. [27] - rx_w_ext_match4, indicates the received frame was matched externally by the ext_match4 input pin. [26] - rx_w_ext_match3, indicates the received frame was matched externally by the ext_match3 input pin. [25] - rx_w_ext_match2, indicates the received frame was matched externally by the ext_match2 input pin. [24] - rx_w_ext_match1, indicates the received frame was matched externally by the ext_match1 input pin.

		[23] - rx_w_uni_hash_match, indicates the received frame was matched as a unicast hash frame. [22] - rx_w_mult_hash_match, indicates the received frame was matched as a multicast hash frame. [21] - rx_w_broadcast_frame, indicates the received frame is a broadcast frame. [20] - rx_w_prty_tagged, indicates a VLAN priority tag detected with received packet. [19:16] - rx_w_tci[3:0], indicates VLAN priority of received packet. [15] - rx_w_vlan_tagged, indicates VLAN tag detected with received packet. [14] - rx_w_bad_frame, indicates received packet is bad. [13:0] - rx_w_frame_length, indicates number of bytes in received packet.
add_match_vec[X:0]	O	These outputs indicate matches for all configured specific address registers. 'X' indicates the number of configured specific address filters. Matches for registers 1 to 4 are also indicated in the rx_w_status vector described above.
rx_w_mod[3:0]	O	Write module, indicates how many bytes are valid on rx_w_data during the last transfer of the packet: 0000- all bytes are valid 0001 - rx_r_data[7:0] is valid 0010 - rx_r_data[15:0] is valid 0011 - rx_r_data[23:0] is valid, and so on until 1111 - rx_r_data[119:0] is valid This output is only valid when coincident with both rx_w_wr and rx_w_eop being high.
rx_w_err	O	Error, active high output indicating the current packet contains an error. This signal is only valid when rx_w_wr is active high and may be set at any time during packet transfer.
rx_w_overflow	I	FIFO overflow, indicates to the MAC that the external RX FIFO has overflowed. The MAC uses this signal for status reporting at the end of frame.
rx_w_flush	O	FIFO flush, active high output indicating that the external RX FIFO should be cleared. This signal is set when the receive path is disabled.
rx_w_timestamp[77:0]	O	Timestamp value at sop for receive frame 48b seconds, 30b nanoseconds. Valid at rx_w_eop.

AMBA (APB) Interface

Signal Name	I/O	Description

`n_preset`	I	Active low AMBA reset (`nPRESET`). This signal must be asserted low asynchronously, and deasserted high synchronously with `pclk`. Resets all APB registers and the `pclk_syncs` block.
`pclk`	I	Peripheral bus clock (`PCLK`).
`psel`	I	Peripheral select (`PSEL`). Active high select signal to indicate that a valid access is being made to one of the GEM_GXL's registers.
`penable`	I	Peripheral enable (`PENABLE`). This indicates the second clock cycle of an access and indicates that the write data may be strobed into a register on the next rising edge of `pclk`, or that the read data is expected to be valid at the next rising edge of `pclk`.
`pwrite`	I	Peripheral write strobe (`PWRITE`). This indicates that a write access is taking place (if `psel` is active).
`paddr[11:2]`	I	Address bus from selected master (`PADDR`). Indicates which register is being accessed. This address is word-aligned within the GEM_GXL and therefore should be connected to bits `[11:02]` of the APB address bus.
`pwdata[31:0]`	I	Write data. Data to be written into the addressed register.
`prdata[31:0]`	O	Read data. Data read from the addressed register. For APB Rev 2.0, it is driven to logic 0 when not addressed.
`perr`	O	Indicates an APB access to an invalid address. Should be sampled when `penable` is high. This signal is optional and not a standard AMBA signal.

AMBA (AHB) Interface

Signal Name	I/O	Description
`n_hreset`	I	Active low asynchronous reset for the DMA. This signal must be asserted low asynchronously and deasserted high synchronously with `hclk`.
`hclk`	I	AHB bus clock.
`hready`	I	Slave device drives this low to extend data phase.
`hresp[1:0]`	I	Slave response, 00 for OK, any other response is not OK and triggers an interrupt. AMBA split and retry operations are not supported.
`hgrant`	I	AHB bus grant.
`haddr[63:0]`	O	AHB address, configurable for 32 bit and 64 bit bus widths.
`htrans[1:0]`	O	AHB transfer type as follows: 00 - Idle 01 - Busy (not used by the GEM) 10 - Non-sequential 11 - Sequential.

`hwrite`	O	High for AHB write, low for AHB read.
`hrdata[127:0]`	I	Data read from memory, configurable for 32 bit, 64 bit or 128 bit bus widths. Unused bits should be tied to logic 0 or logic 1.
`hsize[2:0]`	O	Transfer size, configured according to AMBA AHB data bus width as follows: 010 - 32 bit transfers 011 - 64 bit transfers 100 - 128 bit transfers.
`hburst[2:0]`	O	Burst type, 000 for single transfer, 001 for incrementing burst and 011 for four beat incrementing burst.
`hprot[3:0]`	O	Protection type. Parameterizable via `` `gem_hprot_value ``
`hwdata[127:0]`	O	Data written to memory, configurable for 32 bit, 64 bit or 128 bit bus widths.
`hbusreq`	O	AHB bus request.
`hlock`	O	Always asserted with `hbusreq` to lock the grant. This signal may be left unconnected if required (refer to user guide section 2)

AMBA (AXI4) Interface

Signal Name	I/O	Description
`n_areset`	I	Active low asynchronous reset for the DMA. This signal must be asserted low asynchronously and deasserted high synchronously with `aclk`.
`aclk`	I	AXI bus clock.
`awaddr[63:0]`	O	AXI write channel address, configurable for 32 bit and 64 bit bus widths.
`awlen[7:0]`	O	AXI write channel burst length. Maximum value is 255.
`awsize[2:0]`	O	Transfer size, configured according to AXI data bus width as follows: 010 - 32 bit transfers (ie 4 byte) 011 - 64 bit transfers (ie 8 byte) 100 - 128 bit transfers (ie 16 byte).
`awburst[1:0]`	O	All bursts generated by the AXI DMA are incrementing, so this output is tied to 2'b01.
`awvalid`	O	AXI write request available from DMA.
`awready`	I	AXI write request accepted by slave.
`awid[3:0]`	O	Not currently used by the GEM_GXL DMA. This output is tied low.
`awcache[3:0]`	O	Cache support. Parameterizable via `` `gem_axi_awcache_value ``

`awlock[1:0]`	O	Not currently used by the GEM_GXL DMA. This output is tied low.
`awprot[2:0]`	O	Protection unit support. Parameterizable via `` `gem_axi_prot_value ``
`wdata[127:0]`	O	Data write to memory, configurable for 32 bit, 64 bit or 128 bit bus widths. Unused bits are tied to logic 0.
`wstrb[15:0]`	O	Byte write strobes to memory, configurable for 32 bit, 64 bit or 128 bit data bus widths. Unused bits are tied to logic 0.
`wlast`	O	Identifies the last data of a burst.
`wvalid`	O	AXI write data valid.
`wready`	I	AXI write data accepted by slave.
`bresp[1:0]`	I	Slave response, 00 for OK, any other response is not OK and triggers an interrupt.
`bid[3:0]`	I	Not currently used by the GEM_GXL DMA.
`bvalid`	I	AXI write response bus valid.
`bready`	O	AXI write response accepted.
`araddr[63:0]`	O	AXI read channel address, configurable for 32 bit and 64 bit bus widths.
`arlen[7:0]`	O	AXI read channel burst length. Maximum value is 255.
`arsize[2:0]`	O	Transfer size, configured according to AXI data bus width as follows: 010 - 32 bit transfers (ie 4 byte) 011 - 64 bit transfers (ie 8 byte) 100 - 128 bit transfers (ie 16 byte).
`arburst[1:0]`	O	All bursts generated by the AXI DMA are incrementing, so this output is tied to 2'b01.
`arvalid`	O	AXI read request available from DMA.
`arready`	I	AXI read request accepted by slave.
`arid[3:0]`	O	Not currently used by the GEM_GXL DMA. This output is tied low.
`arcache[3:0]`	O	Cache support. Parameterizable via ` `` `gem_axi_arcache_value ``
`arlock[1:0]`	O	Not currently used by the GEM_GXL DMA. This output is tied low.
`arprot[2:0]`	O	Protection unit support. Parameterizable via ` `` `gem_axi_prot_value ``

`rdata[127:0]`	I	Data read from memory, configurable for 32 bit, 64 bit or 128 bit bus widths. Unused bits should be tied to logic 0 or logic 1.
`rlast`	I	Identifies the last data of a read burst.
`rresp[1:0]`	I	Slave response, 00 for OK, any other response is not OK and triggers an interrupt.
`rid[3:0]`	I	Not currently used by the GEM_GXL DMA.
`rvalid`	I	AXI read data valid
`rready`	O	AXI read data accepted

Transmit Packet Buffer Memory Interface (Dual Port SRAM Configuration)

Port A of the transmit packet memory should be clocked on `hclk/aclk` and port B should be clocked on `tx_clk`.

Signal Name	I/O	Description
`txdpram_ena`	O	Transmit packet buffer memory Port A chip select
`txdpram_wea`	O	Transmit packet buffer memory Port A write enable (always high)
`txdpram_addra[N:0]`	O	Transmit packet buffer memory Port A word address bus. Address width configurable.
`txdpram_dia[127:0]`	O	Transmit packet buffer memory Port A write data bus. Either 32-bit, 64-bit or 128-bitdepending on `dma_bus_width[1:0]`
`txdpram_enb`	O	Transmit packet buffer memory Port B chip select
`txdpram_web`	O	Transmit packet buffer memory Port B write enable (always low)
`txdpram_addrb[N:0]`	O	Transmit packet buffer memory Port B word address bus. Address width configurable.
`txdpram_dob[127:0]`	I	Transmit packet buffer memory Port B read data bus. Either 32-bit,64-bit or 128-bit depending on `dma_bus_width[1:0]`

Transmit Packet Buffer Memory Interface (Single Port SRAM Configuration)

The transmit packet buffer memory should be clocked from hclk/aclk.

Signal Name	I/O	Description
txspram_we	O	Transmit packet buffer memory write enable
txspram_en	O	Transmit packet buffer memory chip select
txspram_addr`[N:0]`	O	Transmit packet buffer memory word address bus.

		Address width configurable.
txspram_do[127:0]	O	Transmit packet buffer memory read data bus. Either 32-bit,64-bit or 128-bit depending on dma_bus_width[1:0]
txspram_di[127:0]	I	Transmit packet buffer memory write data bus. Either 32-bit,64-bit or 128-bit depending on dma_bus_width[1:0]

Receive Packet Buffer Memory Interface (Dual Port SRAM Configuration)

Port A of the receive packet memory should be clocked on `rx_clk` and port B should be clocked on `hclk/aclk`.

Signal Name	I/O	Description
rxdpram_ena	O	Receive packet buffer memory Port A chip select
rxdpram_wea	O	Receive packet buffer memory Port A write enable (always high)
rxdpram_addra[N:0]	O	Receive packet buffer memory Port A word address bus. Address width configurable.
rxdpram_dia[127:0]	O	Receive packet buffer memory Port A write data bus. Either 32-bit,64-bit or 128-bit depending on dma_bus_width[1:0]
rxdpram_enb	O	Receive packet buffer memory Port B chip select
rxdpram_web	O	Receive packet buffer memory Port B write enable (always low)
rxdpram_addrb[N:0]	O	Receive packet buffer memory Port B word address bus. Address width configurable.
rxdpram_dob[63:0]	I	Receive packet buffer memory Port B read data bus. Either 32-bit,64-bit or 128-bit depending on dma_bus_width[1:0]

Receive Packet Buffer Memory Interface (Single Port SRAM Configuration)

The receive packet buffer memory should be clocked from hclk/aclk.

Signal Name	I/O	Description
rxspram_we	O	Receive packet buffer memory write enable
rxspram_en	O	Receive packet buffer memory chip select
rxspram_addr[N:0]	O	Receive packet buffer memory word address bus. Address width configurable.
rxspram_do[127:0]	O	Receive packet buffer memory read data bus. Either 32-bit,64-bit or 128-bit depending on dma_bus_width[1:0]

rxspram_di[127:0]	I	Receive packet buffer memory write data bus. Either 32-bit,64-bit or 128-bit depending on `dma_bus_width[1:0]`

System Interface

Signal Name	I/O	Description
`n_txreset`	I	Active low `tx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `tx_clk`.
`n_rxreset`	I	Active low `rx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `rx_clk`.
`n_gtxreset`	I	Active low `gtx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `gtx_clk`.
`n_gtx20reset`	I	Active low `gtx20_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `gtx20_clk`.
`n_pcs_rxreset`	I	Active low `pcs_rx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `pcs_rx_clk`.
`n_pcs_rx10reset`	I	Active low `pcs_rx10_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `pcs_rx10_clk`.
`n_rbc0reset`	I	Active low `rbc0` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `rbc0`.
`n_rbc1reset`	I	Active low `rbc1` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `rbc1`.
`n_ntxreset`	I	Active low `n_tx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `n_tx_clk`.
`n_tx_clk`	I	Inverted `tx_clk` used to buffer `tx_clk` domain signals to be fed into `rx_clk` domain within the loopback module.
`n_nrxreset`	I	Active low `n_rx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `n_rx_clk`.
`n_ref_reset`	I	Active low `ref_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `ref_clk`.
`n_tsureset`	I	Active low `tsu_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `tsu_clk`.

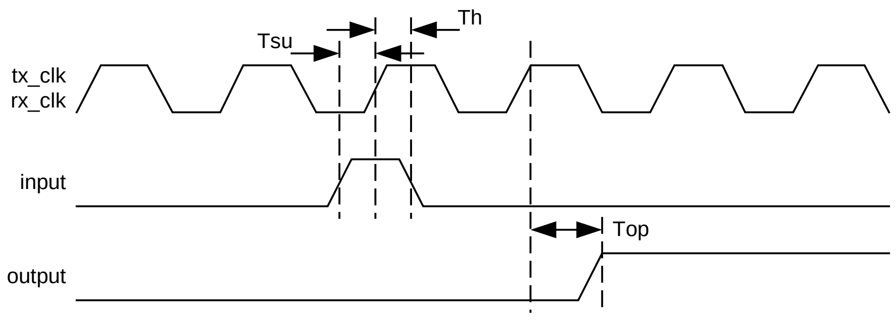
Timing Requirements

For all internal interfaces (including AHB, AXI, APB, FIFO), all inputs must be setup a minimum of 40% of the clock period before the rising edge of the related clock and all outputs will be valid at a maximum of 40% of the clock period after the rising edge of the related clock.

These characteristics have been used as the default values for the trial synthesis of the module using a typical 90nm technology.

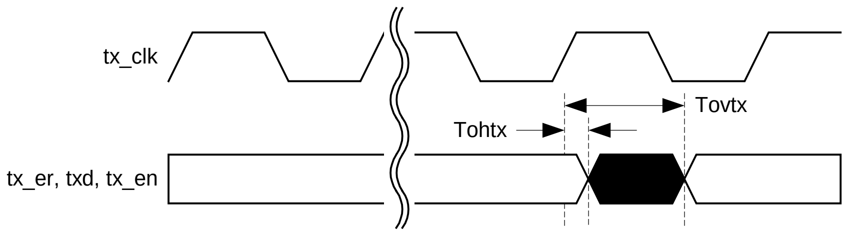
FIFO Interface Timing

Parameter	Description	Min	Max	Unit
Tsu	FIFO setup time	4	—	ns
Th	FIFO hold time	0.5	—	ns
Top	FIFO output time	—	4	ns



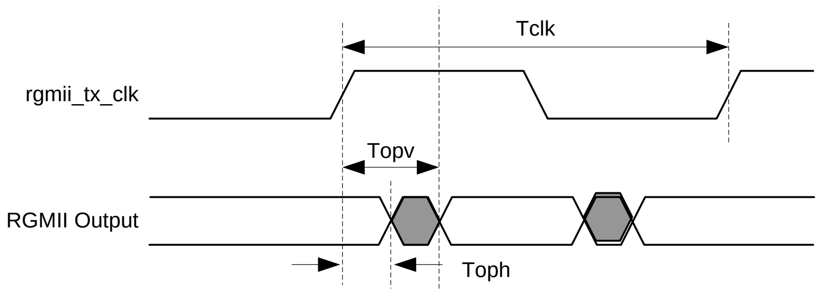
Transmit Timing (MII/GMII)

Parameter	Description	Min	Max	Unit
Tovtx	Transmit data valid after `tx_clk`	-	5.5	ns
Tohtx	Transmit data hold after `tx_clk`	0.5	-	ns



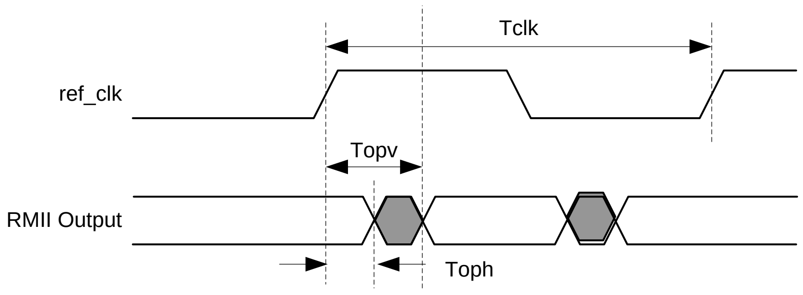
Transmit Timing (RGMII)

Parameter	Description	Min	Max	Unit
Tclk	`tx_clk` clock period	8	DC	ns
Topv	RGMII output data valid after `tx_clk and` `n_tx_clk` output if no internal delay	—	0.5	ns
Toph	RGMII output data hold after `tx_clk` and `n_tx_clk` if no internal delay	-0.5	—	ns



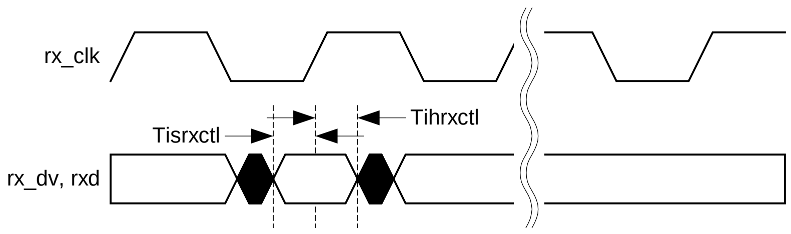
Transmit Timing (RMII)

Parameter	Description	Min	Max	Unit
Tclk	`ref_clk` clock period	20	DC	ns
Topv	RMII output transmit data valid after `ref_clk`	—	5.5	ns
Toph	RMII output transmit data hold after `ref_clk`	0.5	—	ns



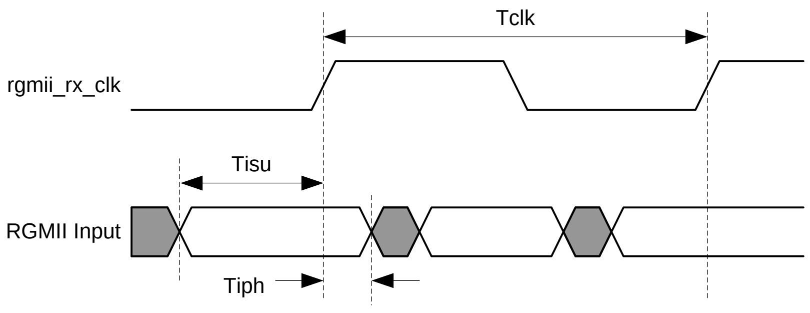
Receive Timing (MII/GMII)

Parameter	Description	Min	Max	Unit
Tisrxctl	Receive data set-up prior to `rx_clk`	2.0	-	ns
Tihrxctl	Receive data hold after `rx_clk`	0	-	ns



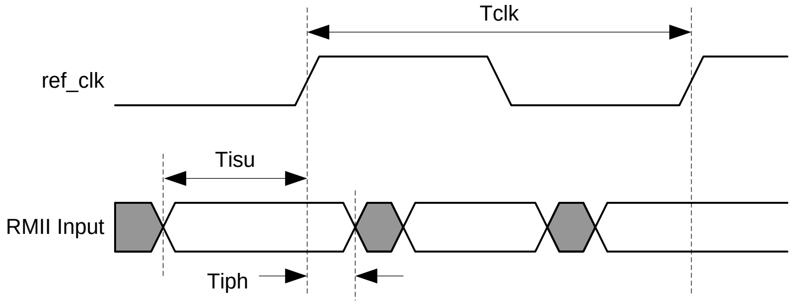
Receive Timing (RGMII)

Parameter	Description	Min	Max	Unit
Tclk	`rx_clk` clock period	8	DC	ns
Tisu	RGMII input data set up prior to `rx_clk`	2.0	—	ns
Tiph	RGMII input data hold after `rx_clk`	0.8	—	ns



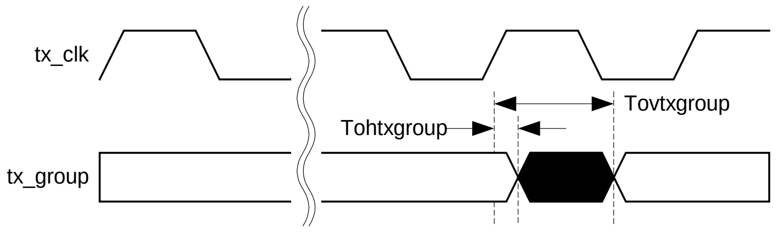
Receive Timing (RMII)

Parameter	Description	Min	Max	Unit
Tclk	`ref_clk` clock period	20	DC	ns
Tisu	RMII input set up prior to `ref_clk`	2.5	—	ns
Tiph	RMII input data hold after `ref_clk`	2	—	ns



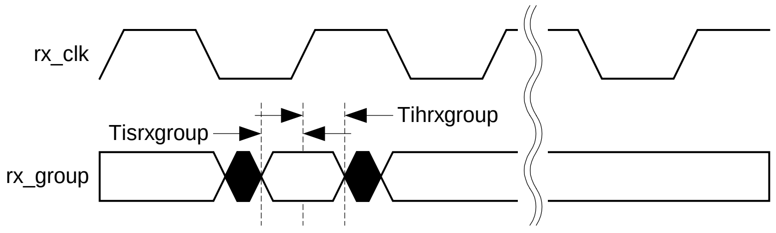
Transmit Timing (TBI)

Parameter	Description	Min	Max	Unit
Tovtxgroup	Transmit data valid after `tx_clk`	-	6.0	ns
Tohtxgroup	Transmit data hold after `tx_clk`	1.0	-	ns



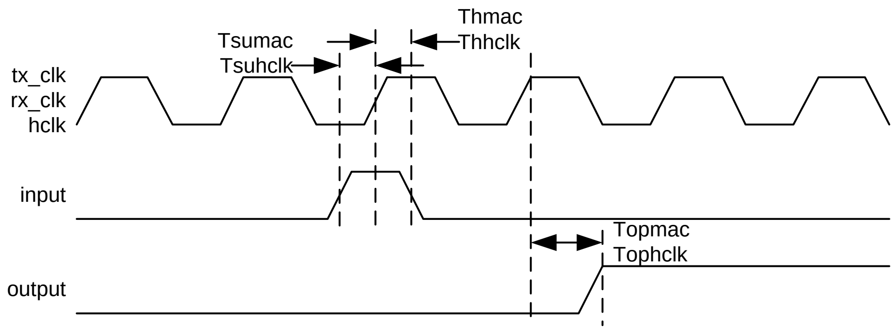
Receive Timing (TBI)

Parameter	Description	Min	Max	Unit
Tisrxgroup	Receive data set-up prior to `rbc0/rbc1`	2.5	-	ns
Tihrxgroup	Receive data hold after `rbc0/rbc1`	-	1.5	ns



Packet Buffer Memory Interface Timing

Parameter	Description	Min	Max	Unit
Tsumac	RX ports A and B or TX port B input setup times	4	—	ns
Thmac	RX ports A and B or TX port B input hold time	0.5	—	ns
Topmac	RX ports A and B or TX port B output time	—	4	ns
Tsuhclk	TX port A input setup times	4	—	ns
Thhclk	TX port A input hold time	0.5	—	ns
Tophclk	TX port A output time	—	4	ns



Clock Domains

Depending on the configuration selected the GEM_GXL has up to nine clock domains used internally within the design. These are described in the following table:

Clock	Speed	Description
`hclk`	10 to 500 MHz	AMBA AHB clock used by the DMA block.
`aclk`	10 to 500 MHz	AMBA AXI clock used by the DMA block.
`pclk`	10 to 500 MHz	AMBA APB clock used by the MAC register block and PCS register block. If the MAC is operated at 2.5G data-rates then the minimum value for pclk becomes 20 MHz.
`tsu_clk`	5 to 400 MHz	Alternate clock source for the time stamp unit.

`tx_clk`	1.25, 2.5,12.5, 25, or 125 MHz	MAC transmit clock, used by the MAC transmit block. In 10/100 GMII mode, `tx_clk` will run at either 2.5 MHz or 25 MHz as determined by the external PHY MII clock input. When using gigabit mode, the transmit clock must be sourced from a 125 MHz reference clock. Depending upon system architecture, this reference clock may be sourced from an on-chip clock multiplier, generated directly from an off-chip oscillator, or taken from the PHY `rx_clk`. In RMII mode, `tx_clk` should be connected to the rmii_tx_clk outputs. In SGMII mode this clock is sourced from gtx_clk. In 100Mb/s SGMII mode gtx_clk phases need to be deleted to bring the effective frequency down to 12.5MHz and in 10Mb/s SGMII down to 1.25MHz.
`gtx_clk`	125 MHz	PCS transmit clock. In non-SGMII applications this may be sourced directly from tx_clk. In SGMII applications this is sourced from the SerDes and divided down in 10 and 100 Mb/s modes to drive tx_clk.
`gtx20_clk`	62.5MHz	PCS transmit clock used at the PCS to PHY interface to support a 20-bit PHY interface.
`rx_clk`	1.25, 2.5, 12.5, 25, 62.5, or 125 MHz	Clock used by the MAC receive synchronization block. In 10/100 and gigabit mode using the GMII/MII interface, this clock is sourced from the `rx_clk` input of the external PHY and will be either 2.5 MHz, 25 MHz or 125 MHz. In RMII modes, rx_clk should be connected to the rmii_rx_clk outputs. When the PCS is selected in gigabit mode, the clock must be sourced from either the `rbc1` or pcs_rx_clk and will be 62.5MHz for Gigabit PHYs. In 10/100Mb/s SGMII mode the clock needs to be divided down to bring the effective frequency down to 12.5MHz in 100Mb/s mode and in 10Mb/s SGMII down to 1.25MHz. In 10/100 SGMII 8 bits of data are transferred from the PCS to the MAC each clock cycle rather than 16.
`rbc0` `rbc1`	Two 62.5 MHz clocks 180° out of phase	Clocks used in the PCS receive channel.
`pcs_rx_clk`	62.5MHz	Clocks used in the PCS receive channel.
`pcs_rx10_clk`	125MHz	Clocks used at the PCS to PHY interface to support a 10-bit PHY interface.

`ref_clk`	50MHz	50MHz reference clock present only when the RMII configuration option is selected. Both the `tx_clk` and `rx_clk` are internally generated from `ref_clk` within the RMII interface. In order to provide maximum flexibility to the user, these generated clocks are fed out and then back into the design. This allows the user to determine their own scan insertion technique and helps with clock tree insertion and balancing. These signals can optionally be multiplexed with an external clock to provide clocking in internal loopback mode. `gem_clk_ctrl.v` gives an example of how this can be done.
`n_tx_clk`	2.5, 25, or 125 MHz	Inverted `tx_clk` used for RGMII and also for the loopback module. For loopback `tx_clk` domain signals are re-timed to this clock before being passed to the `rx_clk` domain receiver inputs
`n_rx_clk`	2.5, 25, or 125 MHz	Inverted `rx_clk` used for RGMII.

The GEM_GXL contains handshaking logic and synchronizers that ensure reliable propagation of signals across clock boundaries.

As far as the GEM_GXL is concerned, management data clock (MDC) is not a clock, but a `pclk` timed output from the registers block.

The maximum frequency of `hclk, aclk` and `pclk` is determined by the speed of the technology library.

The `hclk and aclk` clock frequency must be chosen such that the bandwidth requirements of the chosen transmit and receive Ethernet data rates are met.

When using internal loopback mode, `tx_clk` and `rx_clk` must be provided using the same loopback reference clock, and `n_tx_clk` must be provided with the inverted version of the loopback reference clock.

It is important that receive and transmit are disabled when making the switch into and out of internal loopback, as the clocks provided may glitch whilst switching to the loopback reference clock. Also TBI mode must be disabled for internal loopback. This is because TBI mode configures the receive path to be 16 bits and the transmit path to 8 bits wide.

When operating at gigabit speed using the GMII interface, the system must provide a 125 MHz reference clock to the PHY, normally termed `g_tx_clk`. This clock must be the same clock used for the `tx_clk` input of the GEM to maintain correct relationship between the reference clock and data on the GMII.

Programming Interface

The following registers may be programmed for the GEM_GXL module.

Note: *The Offset Address in the following table is byte-aligned as required by the CPU. It is therefore four times that supplied on `paddr[11:2]`.*

Register Map

Register Name	Type	Width	Reset Value	Address Offset
network_control	RW	32	0x0000 0000	0x000
network_config	RW	32	0x0008 0000	0x004
network_status	RO	32	0x0000 0004	0x008
user_io_register	RW	32	0x0000 0000	0x00C
dma_config	RW	32	0x0002 07C4	0x010
transmit_status	RW	32	0x0000 0000	0x014
receive_q_ptr	RW	32	0x0000 0000	0x018
transmit_q_ptr	RW	32	0x0000 0000	0x01C
receive_status	RW	32	0x0000 0000	0x020
int_status	RW	32	0x0000 0000	0x024
int_enable	RW	32	0x0000 0000	0x028
int_disable	RW	32	0x0000 0000	0x02C
int_mask	RO	32	0x3FFF FFFF	0x030
phy_management	RW	32	0x0000 0000	0x034
pause_time	RO	32	0x0000 0000	0x038
tx_pause_quantum	RW	32	0xFFFF FFFF	0x03C
pbuf_txcutthru	RW	32	0x0000 3FFF	0x040
pbuf_rxcutthru	RW	32	0x0000 07FF	0x044
jumbo_max_length	RW	32	0x0000 2800	0x048
external_fifo_interface	RW	32	0x0000 0000	0x04C
axi_max_pipeline	RW	32	0x0000 0101	0x054
rsc_control	RW	32	0x0000 0000	0x058
int_moderation	RW	32	0x0000 0000	0x05C
sys_wake_time	RW	32	0x0000 0000	0x060
hash_bottom	RW	32	0x0000 0000	0x080
hash_top	RW	32	0x0000 0000	0x084
spec_add1_bottom	RW	32	0x0000 0000	0x088
spec_add1_top	RW	32	0x0000 0000	0x08C
spec_add2_bottom	RW	32	0x0000 0000	0x090
spec_add2_top	RW	32	0x0000 0000	0x094
spec_add3_bottom	RW	32	0x0000 0000	0x098
spec_add3_top	RW	32	0x0000 0000	0x09C
spec_add4_bottom	RW	32	0x0000 0000	0x0A0
spec_add4_top	RW	32	0x0000 0000	0x0A4
spec_type1	RW	32	0x0000 0000	0x0A8
spec_type2	RW	32	0x0000 0000	0x0AC
spec_type3	RW	32	0x0000 0000	0x0B0
spec_type4	RW	32	0x0000 0000	0x0B4
wol_register	RW	32	0x0000 0000	0x0B8
stretch_ratio	RW	32	0x0000 0000	0x0BC
stacked_vlan	RW	32	0x0000 0000	0x0C0
tx_pfc_pause	RW	32	0x0000 0000	0x0C4
mask_add1_bottom	RW	32	0x0000 0000	0x0C8
mask_add1_top	RW	32	0x0000 0000	0x0CC
dma_addr_or_mask	RW	32	0x0000 0000	0x0D0
rx_ptp_unicast	RW	32	0x0000 0000	0x0D4
tx_ptp_unicast	RW	32	0x0000 0000	0x0D8
tsu_nsec_cmp	RW	32	0x0000 0000	0x0DC
tsu_sec_cmp	RW	32	0x0000 0000	0x0E0
tsu_msb_sec_cmp	RW	32	0x0000 0000	0x0E4
tsu_ptp_tx_msb_sec	RO	32	0x0000 0000	0x0E8
tsu_ptp_rx_msb_sec	RO	32	0x0000 0000	0x0EC
tsu_peer_tx_msb_sec	RO	32	0x0000 0000	0x0F0
tsu_peer_rx_msb_sec	RO	32	0x0000 0000	0x0F4
dpram_fill_dbg	RW	32	0x0000 0000	0x0F8
revision_reg	RO	32	0x0007 0100	0x0FC
octets_txed_bottom	RO	32	0x0000 0000	0x100

octets_txed_top	RO	32	0x0000 0000	0x104
frames_txed_ok	RO	32	0x0000 0000	0x108
broadcast_txed	RO	32	0x0000 0000	0x10C
multicast_txed	RO	32	0x0000 0000	0x110
pause_frames_txed	RO	32	0x0000 0000	0x114
frames_txed_64	RO	32	0x0000 0000	0x118
frames_txed_65	RO	32	0x0000 0000	0x11C
frames_txed_128	RO	32	0x0000 0000	0x120
frames_txed_256	RO	32	0x0000 0000	0x124
frames_txed_512	RO	32	0x0000 0000	0x128
frames_txed_1024	RO	32	0x0000 0000	0x12C
frames_txed_1519	RO	32	0x0000 0000	0x130
tx_underruns	RO	32	0x0000 0000	0x134
single_collisions	RO	32	0x0000 0000	0x138
multiple_collisions	RO	32	0x0000 0000	0x13C
excessive_collisions	RO	32	0x0000 0000	0x140
late_collisions	RO	32	0x0000 0000	0x144
deferred_frames	RO	32	0x0000 0000	0x148
crs_errors	RO	32	0x0000 0000	0x14C
octets_rxed_bottom	RO	32	0x0000 0000	0x150
octets_rxed_top	RO	32	0x0000 0000	0x154
frames_rxed_ok	RO	32	0x0000 0000	0x158
broadcast_rxed	RO	32	0x0000 0000	0x15C
multicast_rxed	RO	32	0x0000 0000	0x160
pause_frames_rxed	RO	32	0x0000 0000	0x164
frames_rxed_64	RO	32	0x0000 0000	0x168
frames_rxed_65	RO	32	0x0000 0000	0x16C
frames_rxed_128	RO	32	0x0000 0000	0x170
frames_rxed_256	RO	32	0x0000 0000	0x174
frames_rxed_512	RO	32	0x0000 0000	0x178
frames_rxed_1024	RO	32	0x0000 0000	0x17C
frames_rxed_1519	RO	32	0x0000 0000	0x180
undersize_frames	RO	32	0x0000 0000	0x184
excessive_rx_length	RO	32	0x0000 0000	0x188
rx_jabbers	RO	32	0x0000 0000	0x18C
fcs_errors	RO	32	0x0000 0000	0x190
rx_length_errors	RO	32	0x0000 0000	0x194
rx_symbol_errors	RO	32	0x0000 0000	0x198
alignment_errors	RO	32	0x0000 0000	0x19C
rx_resource_errors	RO	32	0x0000 0000	0x1A0
rx_overruns	RO	32	0x0000 0000	0x1A4
rx_ip_ck_errors	RO	32	0x0000 0000	0x1A8
rx_tcp_ck_errors	RO	32	0x0000 0000	0x1AC
rx_udp_ck_errors	RO	32	0x0000 0000	0x1B0
auto_flushed_pkts	RO	32	0x0000 0000	0x1B4
tsu_timer_incr_sub_nsec	RW	32	0x0000 0000	0x1BC
tsu_timer_msb_sec	RW	32	0x0000 0000	0x1C0
tsu_strobe_msb_sec	RO	32	0x0000 0000	0x1C4
tsu_strobe_sec	RO	32	0x0000 0000	0x1C8
tsu_strobe_nsec	RO	32	0x0000 0000	0x1CC
tsu_timer_sec	RW	32	0x0000 0000	0x1D0
tsu_timer_nsec	RW	32	0x0000 0000	0x1D4
tsu_timer_adjust	RW	32	0x0000 0000	0x1D8
tsu_timer_incr	RW	32	0x0000 0000	0x1DC
tsu_ptp_tx_sec	RO	32	0x0000 0000	0x1E0
tsu_ptp_tx_nsec	RO	32	0x0000 0000	0x1E4
tsu_ptp_rx_sec	RO	32	0x0000 0000	0x1E8
tsu_ptp_rx_nsec	RO	32	0x0000 0000	0x1EC
tsu_peer_tx_sec	RO	32	0x0000 0000	0x1F0
tsu_peer_tx_nsec	RO	32	0x0000 0000	0x1F4
tsu_peer_rx_sec	RO	32	0x0000 0000	0x1F8

tsu_peer_rx_nsec	RO	32	0x0000 0000	0x1FC
pcs_control	RW	32	0x0000 9040	0x200
pcs_status	RO	32	0x0000 0109	0x204
pcs_an_adv	RW	32	0x0000 0020	0x210
pcs_an_lp_base	RO	32	0x0000 0000	0x214
pcs_an_exp	RO	32	0x0000 0004	0x218
pcs_an_np_tx	RW	32	0x0000 0000	0x21C
pcs_an_lp_np	RO	32	0x0000 0000	0x220
pcs_an_ext_status	RO	32	0x0000 8000	0x23C
tx_pause_quantum1	RW	32	0xFFFF FFFF	0x260
tx_pause_quantum2	RW	32	0xFFFF FFFF	0x264
tx_pause_quantum3	RW	32	0xFFFF FFFF	0x268
rx_lpi	RO	32	0x0000 0000	0x270
rx_lpi_time	RO	32	0x0000 0000	0x274
tx_lpi	RO	32	0x0000 0000	0x278
tx_lpi_time	RO	32	0x0000 0000	0x27C
designcfg_debug1	RO	32	0x0498 4310	0x280
designcfg_debug2	RO	32	0x7AF1 2800	0x284
designcfg_debug3	RO	32	0x2400 0000	0x288
designcfg_debug4	RO	32	0x0000 0000	0x28C
designcfg_debug5	RO	32	0x502F A345	0x290
designcfg_debug6	RO	32	0x0754 FFFE	0x294
designcfg_debug7	RO	32	0x0000 0000	0x298
designcfg_debug8	RO	32	0x1010 0820	0x29C
designcfg_debug9	RO	32	0x0000 0000	0x2A0
designcfg_debug10	RO	32	0x2224 4444	0x2A4
spec_add5_bottom	RW	32	0x0000 0000	0x300
spec_add5_top	RW	32	0x0000 0000	0x304
spec_add6_bottom	RW	32	0x0000 0000	0x308
spec_add6_top	RW	32	0x0000 0000	0x30C
spec_add7_bottom	RW	32	0x0000 0000	0x310
spec_add7_top	RW	32	0x0000 0000	0x314
spec_add8_bottom	RW	32	0x0000 0000	0x318
spec_add8_top	RW	32	0x0000 0000	0x31C
spec_add9_bottom	RW	32	0x0000 0000	0x320
spec_add9_top	RW	32	0x0000 0000	0x324
spec_add10_bottom	RW	32	0x0000 0000	0x328
spec_add10_top	RW	32	0x0000 0000	0x32C
spec_add11_bottom	RW	32	0x0000 0000	0x330
spec_add11_top	RW	32	0x0000 0000	0x334
spec_add12_bottom	RW	32	0x0000 0000	0x338
spec_add12_top	RW	32	0x0000 0000	0x33C
spec_add13_bottom	RW	32	0x0000 0000	0x340
spec_add13_top	RW	32	0x0000 0000	0x344
spec_add14_bottom	RW	32	0x0000 0000	0x348
spec_add14_top	RW	32	0x0000 0000	0x34C
spec_add15_bottom	RW	32	0x0000 0000	0x350
spec_add15_top	RW	32	0x0000 0000	0x354
spec_add16_bottom	RW	32	0x0000 0000	0x358
spec_add16_top	RW	32	0x0000 0000	0x35C
spec_add17_bottom	RW	32	0x0000 0000	0x360
spec_add17_top	RW	32	0x0000 0000	0x364
spec_add18_bottom	RW	32	0x0000 0000	0x368
spec_add18_top	RW	32	0x0000 0000	0x36C
spec_add19_bottom	RW	32	0x0000 0000	0x370
spec_add19_top	RW	32	0x0000 0000	0x374
spec_add20_bottom	RW	32	0x0000 0000	0x378
spec_add20_top	RW	32	0x0000 0000	0x37C
spec_add21_bottom	RW	32	0x0000 0000	0x380
spec_add21_top	RW	32	0x0000 0000	0x384
spec_add22_bottom	RW	32	0x0000 0000	0x388

spec_add22_top	RW	32	0x0000 0000	0x38C
spec_add23_bottom	RW	32	0x0000 0000	0x390
spec_add23_top	RW	32	0x0000 0000	0x394
spec_add24_bottom	RW	32	0x0000 0000	0x398
spec_add24_top	RW	32	0x0000 0000	0x39C
spec_add25_bottom	RW	32	0x0000 0000	0x3A0
spec_add25_top	RW	32	0x0000 0000	0x3A4
spec_add26_bottom	RW	32	0x0000 0000	0x3A8
spec_add26_top	RW	32	0x0000 0000	0x3AC
spec_add27_bottom	RW	32	0x0000 0000	0x3B0
spec_add27_top	RW	32	0x0000 0000	0x3B4
spec_add28_bottom	RW	32	0x0000 0000	0x3B8
spec_add28_top	RW	32	0x0000 0000	0x3BC
spec_add29_bottom	RW	32	0x0000 0000	0x3C0
spec_add29_top	RW	32	0x0000 0000	0x3C4
spec_add30_bottom	RW	32	0x0000 0000	0x3C8
spec_add30_top	RW	32	0x0000 0000	0x3CC
spec_add31_bottom	RW	32	0x0000 0000	0x3D0
spec_add31_top	RW	32	0x0000 0000	0x3D4
spec_add32_bottom	RW	32	0x0000 0000	0x3D8
spec_add32_top	RW	32	0x0000 0000	0x3DC
spec_add33_bottom	RW	32	0x0000 0000	0x3E0
spec_add33_top	RW	32	0x0000 0000	0x3E4
spec_add34_bottom	RW	32	0x0000 0000	0x3E8
spec_add34_top	RW	32	0x0000 0000	0x3EC
spec_add35_bottom	RW	32	0x0000 0000	0x3F0
spec_add35_top	RW	32	0x0000 0000	0x3F4
spec_add36_bottom	RW	32	0x0000 0000	0x3F8
spec_add36_top	RW	32	0x0000 0000	0x3FC
int_q1_status	RO	32	0x0000 0000	0x400
int_q2_status	RO	32	0x0000 0000	0x404
int_q3_status	RO	32	0x0000 0000	0x408
int_q4_status	RO	32	0x0000 0000	0x40C
int_q5_status	RO	32	0x0000 0000	0x410
int_q6_status	RO	32	0x0000 0000	0x414
int_q7_status	RO	32	0x0000 0000	0x418
int_q8_status	RO	32	0x0000 0000	0x41C
int_q9_status	RO	32	0x0000 0000	0x420
int_q10_status	RO	32	0x0000 0000	0x424
int_q11_status	RO	32	0x0000 0000	0x428
int_q12_status	RO	32	0x0000 0000	0x42C
int_q13_status	RO	32	0x0000 0000	0x430
int_q14_status	RO	32	0x0000 0000	0x434
int_q15_status	RO	32	0x0000 0000	0x438
transmit_q1_ptr	RW	32	0x0000 0000	0x440
transmit_q2_ptr	RW	32	0x0000 0000	0x444
transmit_q3_ptr	RW	32	0x0000 0000	0x448
transmit_q4_ptr	RW	32	0x0000 0000	0x44C
transmit_q5_ptr	RW	32	0x0000 0000	0x450
transmit_q6_ptr	RW	32	0x0000 0000	0x454
transmit_q7_ptr	RW	32	0x0000 0000	0x458
transmit_q8_ptr	RW	32	0x0000 0000	0x45C
transmit_q9_ptr	RW	32	0x0000 0000	0x460
transmit_q10_ptr	RW	32	0x0000 0000	0x464
transmit_q11_ptr	RW	32	0x0000 0000	0x468
transmit_q12_ptr	RW	32	0x0000 0000	0x46C
transmit_q13_ptr	RW	32	0x0000 0000	0x470
transmit_q14_ptr	RW	32	0x0000 0000	0x474
transmit_q15_ptr	RW	32	0x0000 0000	0x478
receive_q1_ptr	RW	32	0x0000 0000	0x480
receive_q2_ptr	RW	32	0x0000 0000	0x484

receive_q3_ptr	RW	32	0x0000 0000	0x488
receive_q4_ptr	RW	32	0x0000 0000	0x48C
receive_q5_ptr	RW	32	0x0000 0000	0x490
receive_q6_ptr	RW	32	0x0000 0000	0x494
receive_q7_ptr	RW	32	0x0000 0000	0x498
dma_rxbuf_size_q1	RW	32	0x0000 0002	0x4A0
dma_rxbuf_size_q2	RW	32	0x0000 0002	0x4A4
dma_rxbuf_size_q3	RW	32	0x0000 0002	0x4A8
dma_rxbuf_size_q4	RW	32	0x0000 0002	0x4AC
dma_rxbuf_size_q5	RW	32	0x0000 0002	0x4B0
dma_rxbuf_size_q6	RW	32	0x0000 0002	0x4B4
dma_rxbuf_size_q7	RW	32	0x0000 0002	0x4B8
cbs_control	RW	32	0x0000 0000	0x4BC
cbs_idleslope_q_a	RW	32	0x0000 0000	0x4C0
cbs_idleslope_q_b	RW	32	0x0000 0000	0x4C4
upper_tx_q_base_addr	RW	32	0x0000 0000	0x4C8
tx_bd_control	RW	32	0x0000 0000	0x4CC
rx_bd_control	RW	32	0x0000 0000	0x4D0
upper_rx_q_base_addr	RW	32	0x0000 0000	0x4D4
screening_type_1_register_0	RW	32	0x0000 0000	0x500
screening_type_1_register_1	RW	32	0x0000 0000	0x504
screening_type_1_register_2	RW	32	0x0000 0000	0x508
screening_type_1_register_3	RW	32	0x0000 0000	0x50C
screening_type_1_register_4	RW	32	0x0000 0000	0x510
screening_type_1_register_5	RW	32	0x0000 0000	0x514
screening_type_1_register_6	RW	32	0x0000 0000	0x518
screening_type_1_register_7	RW	32	0x0000 0000	0x51C
screening_type_1_register_8	RW	32	0x0000 0000	0x520
screening_type_1_register_9	RW	32	0x0000 0000	0x524
screening_type_1_register_10	RW	32	0x0000 0000	0x528
screening_type_1_register_11	RW	32	0x0000 0000	0x52C
screening_type_1_register_12	RW	32	0x0000 0000	0x530
screening_type_1_register_13	RW	32	0x0000 0000	0x534
screening_type_1_register_14	RW	32	0x0000 0000	0x538
screening_type_1_register_15	RW	32	0x0000 0000	0x53C
screening_type_2_register_0	RW	32	0x0000 0000	0x540
screening_type_2_register_1	RW	32	0x0000 0000	0x544
screening_type_2_register_2	RW	32	0x0000 0000	0x548
screening_type_2_register_3	RW	32	0x0000 0000	0x54C
screening_type_2_register_4	RW	32	0x0000 0000	0x550
screening_type_2_register_5	RW	32	0x0000 0000	0x554
screening_type_2_register_6	RW	32	0x0000 0000	0x558
screening_type_2_register_7	RW	32	0x0000 0000	0x55C
screening_type_2_register_8	RW	32	0x0000 0000	0x560
screening_type_2_register_9	RW	32	0x0000 0000	0x564
screening_type_2_register_10	RW	32	0x0000 0000	0x568
screening_type_2_register_11	RW	32	0x0000 0000	0x56C
screening_type_2_register_12	RW	32	0x0000 0000	0x570
screening_type_2_register_13	RW	32	0x0000 0000	0x574
screening_type_2_register_14	RW	32	0x0000 0000	0x578
screening_type_2_register_15	RW	32	0x0000 0000	0x57C
tx_sched_ctrl	RW	32	0x0000 0000	0x580
bw_rate_limit_q0to3	RW	32	0x0000 0000	0x590
bw_rate_limit_q4to7	RW	32	0x0000 0000	0x594
bw_rate_limit_q8to11	RW	32	0x0000 0000	0x598
bw_rate_limit_q12to15	RW	32	0x0000 0000	0x59C
tx_q_seg_alloc_q0to7	RW	32	0x0000 0000	0x5A0
tx_q_seg_alloc_q8to15	RW	32	0x0000 0000	0x5A4
receive_q8_ptr	RW	32	0x0000 0000	0x5C0
receive_q9_ptr	RW	32	0x0000 0000	0x5C4
receive_q10_ptr	RW	32	0x0000 0000	0x5C8

receive_q11_ptr	RW	32	0x0000 0000	0x5CC
receive_q12_ptr	RW	32	0x0000 0000	0x5D0
receive_q13_ptr	RW	32	0x0000 0000	0x5D4
receive_q14_ptr	RW	32	0x0000 0000	0x5D8
receive_q15_ptr	RW	32	0x0000 0000	0x5DC
dma_rxbuf_size_q8	RW	32	0x0000 0002	0x5E0
dma_rxbuf_size_q9	RW	32	0x0000 0002	0x5E4
dma_rxbuf_size_q10	RW	32	0x0000 0002	0x5E8
dma_rxbuf_size_q11	RW	32	0x0000 0002	0x5EC
dma_rxbuf_size_q12	RW	32	0x0000 0002	0x5F0
dma_rxbuf_size_q13	RW	32	0x0000 0002	0x5F4
dma_rxbuf_size_q14	RW	32	0x0000 0002	0x5F8
dma_rxbuf_size_q15	RW	32	0x0000 0002	0x5FC
int_q1_enable	RW	32	0x0000 0000	0x600
int_q2_enable	RW	32	0x0000 0000	0x604
int_q3_enable	RW	32	0x0000 0000	0x608
int_q4_enable	RW	32	0x0000 0000	0x60C
int_q5_enable	RW	32	0x0000 0000	0x610
int_q6_enable	RW	32	0x0000 0000	0x614
int_q7_enable	RW	32	0x0000 0000	0x618
int_q1_disable	RW	32	0x0000 0000	0x620
int_q2_disable	RW	32	0x0000 0000	0x624
int_q3_disable	RW	32	0x0000 0000	0x628
int_q4_disable	RW	32	0x0000 0000	0x62C
int_q5_disable	RW	32	0x0000 0000	0x630
int_q6_disable	RW	32	0x0000 0000	0x634
int_q7_disable	RW	32	0x0000 0000	0x638
int_q1_mask	RO	32	0x0000 08E6	0x640
int_q2_mask	RO	32	0x0000 08E6	0x644
int_q3_mask	RO	32	0x0000 08E6	0x648
int_q4_mask	RO	32	0x0000 08E6	0x64C
int_q5_mask	RO	32	0x0000 08E6	0x650
int_q6_mask	RO	32	0x0000 08E6	0x654
int_q7_mask	RO	32	0x0000 08E6	0x658
int_q8_enable	RW	32	0x0000 0000	0x660
int_q9_enable	RW	32	0x0000 0000	0x664
int_q10_enable	RW	32	0x0000 0000	0x668
int_q11_enable	RW	32	0x0000 0000	0x66C
int_q12_enable	RW	32	0x0000 0000	0x670
int_q13_enable	RW	32	0x0000 0000	0x674
int_q14_enable	RW	32	0x0000 0000	0x678
int_q15_enable	RW	32	0x0000 0000	0x67C
int_q8_disable	RW	32	0x0000 0000	0x680
int_q9_disable	RW	32	0x0000 0000	0x684
int_q10_disable	RW	32	0x0000 0000	0x688
int_q11_disable	RW	32	0x0000 0000	0x68C
int_q12_disable	RW	32	0x0000 0000	0x690
int_q13_disable	RW	32	0x0000 0000	0x694
int_q14_disable	RW	32	0x0000 0000	0x698
int_q15_disable	RW	32	0x0000 0000	0x69C
int_q8_mask	RO	32	0x0000 08E6	0x6A0
int_q9_mask	RO	32	0x0000 08E6	0x6A4
int_q10_mask	RO	32	0x0000 08E6	0x6A8
int_q11_mask	RO	32	0x0000 08E6	0x6AC
int_q12_mask	RO	32	0x0000 08E6	0x6B0
int_q13_mask	RO	32	0x0000 08E6	0x6B4
int_q14_mask	RO	32	0x0000 08E6	0x6B8
int_q15_mask	RO	32	0x0000 08E6	0x6BC
screening_type_2_ethertype_reg_0	RW	32	0x0000 0000	0x6E0
screening_type_2_ethertype_reg_1	RW	32	0x0000 0000	0x6E4
screening_type_2_ethertype_reg_2	RW	32	0x0000 0000	0x6E8

screening_type_2_ethertype_reg_3	RW	32	0x0000 0000	0x6EC
screening_type_2_ethertype_reg_4	RW	32	0x0000 0000	0x6F0
screening_type_2_ethertype_reg_5	RW	32	0x0000 0000	0x6F4
screening_type_2_ethertype_reg_6	RW	32	0x0000 0000	0x6F8
screening_type_2_ethertype_reg_7	RW	32	0x0000 0000	0x6FC
type2_compare_0_word_0	RW	32	0x0000 0000	0x700
type2_compare_0_word_1	RW	32	0x0000 0000	0x704
type2_compare_1_word_0	RW	32	0x0000 0000	0x708
type2_compare_1_word_1	RW	32	0x0000 0000	0x70C
type2_compare_2_word_0	RW	32	0x0000 0000	0x710
type2_compare_2_word_1	RW	32	0x0000 0000	0x714
type2_compare_3_word_0	RW	32	0x0000 0000	0x718
type2_compare_3_word_1	RW	32	0x0000 0000	0x71C
type2_compare_4_word_0	RW	32	0x0000 0000	0x720
type2_compare_4_word_1	RW	32	0x0000 0000	0x724
type2_compare_5_word_0	RW	32	0x0000 0000	0x728
type2_compare_5_word_1	RW	32	0x0000 0000	0x72C
type2_compare_6_word_0	RW	32	0x0000 0000	0x730
type2_compare_6_word_1	RW	32	0x0000 0000	0x734
type2_compare_7_word_0	RW	32	0x0000 0000	0x738
type2_compare_7_word_1	RW	32	0x0000 0000	0x73C
type2_compare_8_word_0	RW	32	0x0000 0000	0x740
type2_compare_8_word_1	RW	32	0x0000 0000	0x744
type2_compare_9_word_0	RW	32	0x0000 0000	0x748
type2_compare_9_word_1	RW	32	0x0000 0000	0x74C
type2_compare_10_word_0	RW	32	0x0000 0000	0x750
type2_compare_10_word_1	RW	32	0x0000 0000	0x754
type2_compare_11_word_0	RW	32	0x0000 0000	0x758
type2_compare_11_word_1	RW	32	0x0000 0000	0x75C
type2_compare_12_word_0	RW	32	0x0000 0000	0x760
type2_compare_12_word_1	RW	32	0x0000 0000	0x764
type2_compare_13_word_0	RW	32	0x0000 0000	0x768
type2_compare_13_word_1	RW	32	0x0000 0000	0x76C
type2_compare_14_word_0	RW	32	0x0000 0000	0x770
type2_compare_14_word_1	RW	32	0x0000 0000	0x774
type2_compare_15_word_0	RW	32	0x0000 0000	0x778
type2_compare_15_word_1	RW	32	0x0000 0000	0x77C
type2_compare_16_word_0	RW	32	0x0000 0000	0x780
type2_compare_16_word_1	RW	32	0x0000 0000	0x784
type2_compare_17_word_0	RW	32	0x0000 0000	0x788
type2_compare_17_word_1	RW	32	0x0000 0000	0x78C
type2_compare_18_word_0	RW	32	0x0000 0000	0x790
type2_compare_18_word_1	RW	32	0x0000 0000	0x794
type2_compare_19_word_0	RW	32	0x0000 0000	0x798
type2_compare_19_word_1	RW	32	0x0000 0000	0x79C
type2_compare_20_word_0	RW	32	0x0000 0000	0x7A0
type2_compare_20_word_1	RW	32	0x0000 0000	0x7A4
type2_compare_21_word_0	RW	32	0x0000 0000	0x7A8
type2_compare_21_word_1	RW	32	0x0000 0000	0x7AC
type2_compare_22_word_0	RW	32	0x0000 0000	0x7B0
type2_compare_22_word_1	RW	32	0x0000 0000	0x7B4
type2_compare_23_word_0	RW	32	0x0000 0000	0x7B8
type2_compare_23_word_1	RW	32	0x0000 0000	0x7BC
type2_compare_24_word_0	RW	32	0x0000 0000	0x7C0
type2_compare_24_word_1	RW	32	0x0000 0000	0x7C4
type2_compare_25_word_0	RW	32	0x0000 0000	0x7C8
type2_compare_25_word_1	RW	32	0x0000 0000	0x7CC
type2_compare_26_word_0	RW	32	0x0000 0000	0x7D0
type2_compare_26_word_1	RW	32	0x0000 0000	0x7D4
type2_compare_27_word_0	RW	32	0x0000 0000	0x7D8
type2_compare_27_word_1	RW	32	0x0000 0000	0x7DC

type2_compare_28_word_0	RW	32	0x0000 0000	0x7E0
type2_compare_28_word_1	RW	32	0x0000 0000	0x7E4
type2_compare_29_word_0	RW	32	0x0000 0000	0x7E8
type2_compare_29_word_1	RW	32	0x0000 0000	0x7EC
type2_compare_30_word_0	RW	32	0x0000 0000	0x7F0
type2_compare_30_word_1	RW	32	0x0000 0000	0x7F4
type2_compare_31_word_0	RW	32	0x0000 0000	0x7F8
type2_compare_31_word_1	RW	32	0x0000 0000	0x7FC

Physical Estimates

Approximate two input NAND equivalent gate counts:

Feature	**Area**
Basic Configuration	52k
The PCS option adds approximately	10k
The TSU option adds approximately	10k
The GEM statistics registers option adds approximately	32k
The DMA packet buffer (AHB) option adds approximately	100k
The DMA packet buffer (AXI) option adds approximately	142k
The Priority Queue option adds approximately	37k
Each additional DMA priority queue adds (excluding number of screener registers added)	7k
The alternate register based DMA packet buffer (AHB only) option adds	72k

The numbers reported are based on a synthesis and trial place and route flow using a TSMC 28nm HPM process. Gate sizes of 0.3645 um^2 were used to calculate the specific gate count number. This area estimate may significantly change based on gate size, relative size of flipflops and technology used.

As an example, a design with the following configuration had an area of 345k gates using a TSMC 28nm HPM process with a gate size of 0.3645 um^2.

- Basic configuration
- DMA (AXI) option
- GEM statistic registers
- TSU option
- 3 Priority queues
- 32 Screener registers
- AXI Frequency 250MHz
- APB Frequency 100MHz
- AHB Bus Width 64 bits
- Ethernet speed 1Gbps

This example configuration drew a power consumption as follows:

Leakage(mW) = 0.106

Net(mW) = 4.298

Dynamic(mW) = 1.011

Total(mW) = 5.309

Section 2 - Programming the GEM_GXL

Introduction

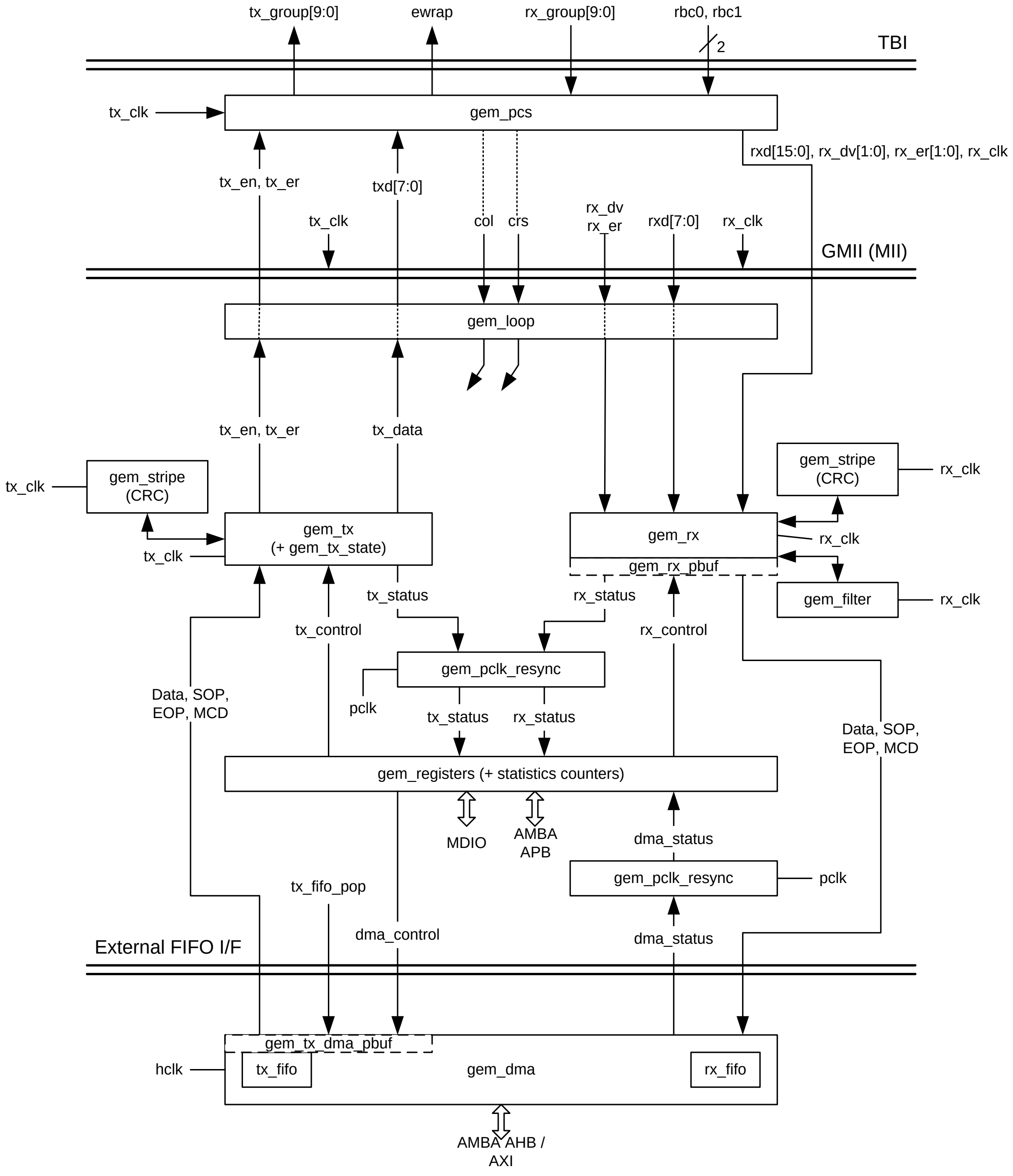
This section enables the GEM_GXL module to be programmed as part of an SoC design. It includes:

Functional description and operation

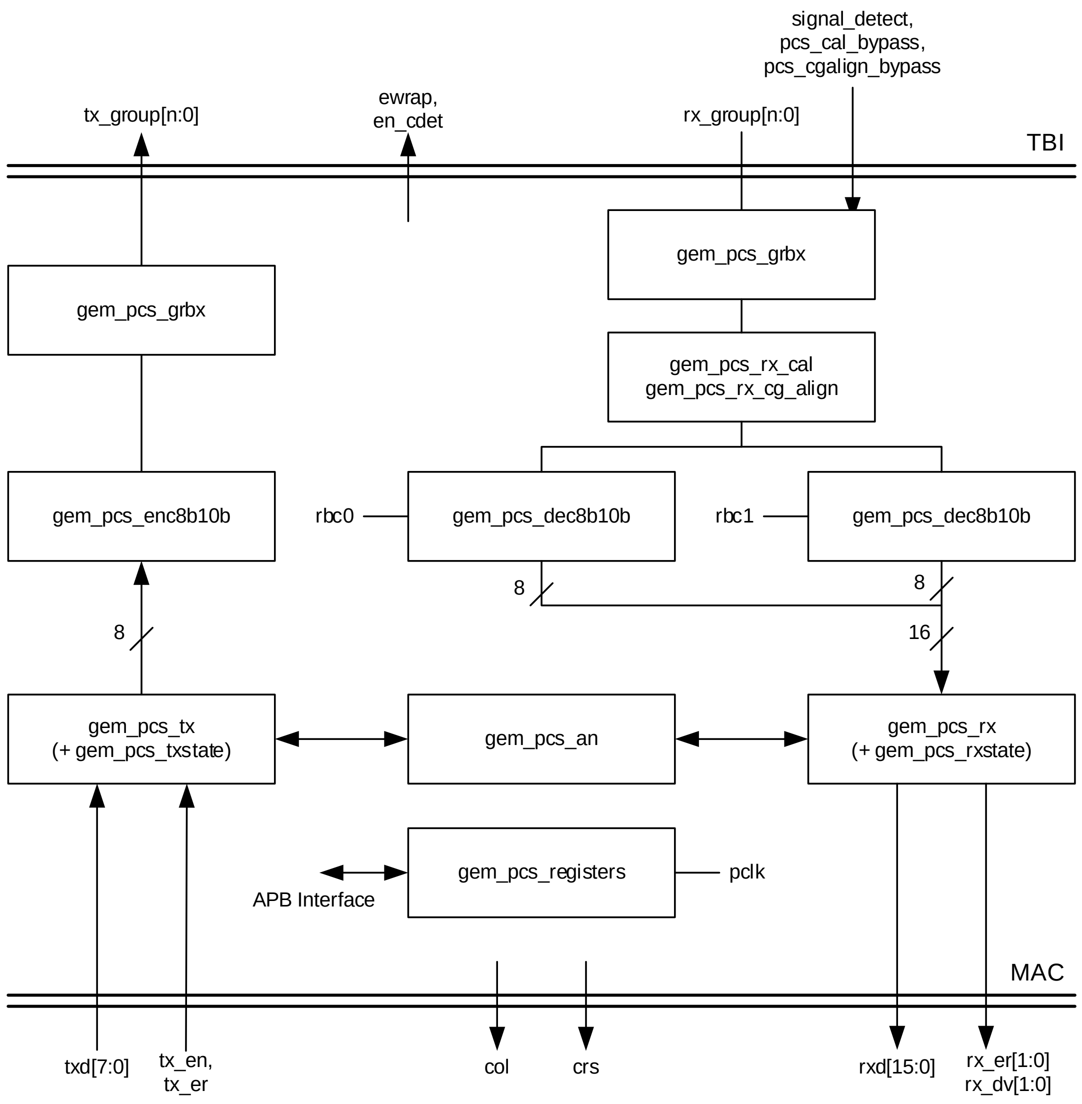
Register map

Programming interface

MAC Functional Block Diagram



PCS Functional Block Diagram



Direct Memory Access Interface

GEM_GXL is supplied with an optional DMA interface. When the GEM_GXL is configured to use the DMA, it is attached to the MAC module’s external FIFO interfaces to provide a scatter gather type capability for packet data storage in an embedded processor system or System on Chip. If the GEM_GXL is not configured to use the DMA, then the external FIFO interface is left exposed for connecting to external memory controllers and the DMA is removed to save area.

When included in the GEM_GXL, the DMA can be configured in two possible modes, the first using internal transmit and receive FIFOs implemented as synthesisable flip-flops and the second is a packet buffering mode, where Single-Port or Dual-Port memories are used to buffer multiple frames. Both have identical software and hardware interfaces (although the packet buffer mode reduces software overhead in certain circumstances). The key feature differences are listed below (the details of these features are described in more detail later in this document):

Internal transmit/receive FIFO DMA

AHB support only

Lowest possible latency solution

32, 64 and 128 bit databus width support

Packet Buffer DMA

AHB or AXI4 support

32, 64 and 128 bit databus width support. Note that while it is supported, 32bit databus widths are not recommended when trying to meet 10Gbps data-rates. The host frequency required is very high.

32 or 64 bit address bus width support

Easier to guarantee maximum line rate due to the ability to store multiple frames in the packet buffer

Makes efficient use of the AHB / AXI interface

Support for up to 16 outstanding AXI transactions in flight. These transactions can cross multiple frame transfers providing excellent support for systems with high latency.

Optional local descriptor caching to improve performance for AXI systems with high latency.

Full store and forward, or partial store and forward programmable options (partial will cater for shorter latency requirements). Note. Partial store and forward is not supported for an AHB configuration with more than one priority queue. Partial store and forward mode is however supported for AXI configurations with priority queues.

Support for Transmit TCP/IP checksum offload

Support for priority queuing

Support for TCP/IP advanced offloads to reduce CPU overhead

When a collision on the line occurs during transmission, the packet will be automatically replayed directly from the packet buffer memory rather than having to re-fetch through the AHB (full store and forward ONLY)

Received errored packets are automatically dropped before any of the packet is presented to external memory (full store and forward ONLY), thus reducing AHB or AXI activity

Supports manual RX packet flush capabilities

Optional RX packet flush when there is lack of resource

Optional burst padding at end of packet and end of buffer to maximize AHB / AXI efficiency

Optional 64 bit addressing for Data Buffer start address within Buffer Descriptor Entry.

Optional Tx/ Rx Timestamp capture to Buffer Descriptor Entry,

Dual port or single port SRAM support.

Using the AXI Interface

The highest data throughput and system performance will be met when the GEM_GXL is configured to use the AXI interface. The GEM_GXL's AXI4 interface provides separate data and address connections for Reads and Writes, which allow simultaneous, bidirectional data transfers. The GEM_GXL supports multiple outstanding transactions on both the Read and Write address channels, up to a programmable limit (see AXI MAX PIPELINE programmable register). This means the issuing of Read and Write requests from the GEM_GXL DMA (on AXI AR and AW channels) are decoupled from the AXI slave responses (on R and B channels). The issuing of outstanding transactions are allowed to span multiple frames, maintaining high data transfer rates even with very high latency systems (when the transactions can take a very long time to complete).

To further provide support for high latency systems, the DMA will buffer the descriptor accesses locally to avoid the underlying DMA from pausing while descriptor transactions are completed by the system fabric.

- TX and RX descriptor reads are issued up-front and stored in a local buffer to feed the underlying DMA when required. This optimizes performance and avoids the need for the underlying DMA to pause while new descriptor fetches are sent to the system bus. The size of the descriptor read buffers are configurable. For low latency systems, these buffers can be configured small.
- TX and RX descriptor writes issued by the underlying DMA are buffered locally to avoid holding up the underlying DMA if and when the system delays the completion of descriptor writes. The size of the descriptor write buffers are configurable. For low latency systems, these buffers can be configured small. Note a descriptor write transaction is not considered complete until the 'B' response associated with that transaction has been collected from the fabric.

By providing these descriptor buffers, the DMA does not need to use AXI ID's to support high latency systems. AXI ID's could have been used for example to differentiate between traffic types. This could theoretically ensure that transactions issued by the underlying DMA that require to be completed before normal data-flow can continue are given unique ID's that could be processed out-of-order by the system fabric (the traffic types that would require this would be the traffic associated with descriptor accesses). The use of AXI ID's requires parallel out-of-order completion support in the system fabric and slaves. Any fabric or system limitations with regards to out-of-order support would need to carefully analyzed to ensure system throughput could be guaranteed. The GEM_GXL does not provide AXI ID support and encompasses its own buffering to support high latency systems using a single ID.

The maximum burst lengths the GEM_GXL uses are programmable. Single, bursts up to 4, 8, 16 or 256 can be selected (256 is supported by AXI4 only). With a 64 bit datapath and a burst length setting of 256, up to 2KB transfers can be made with a single request. The burst length is controlled via the DMA configuration register.

The DMA adheres to the AXI4 specification and will never issue transactions that break a 4KB boundary. For example, if the burst length is set to 256, a packet to be transmitted is 100 bytes long, the datapath is 64bits wide, and the data is located in system memory 60 bytes from a 4KB boundary, the GEM_GXL DMA will issue two burst requests, one of length 7 and the other of length 5.

The GEM_GXL provides an AXI4 interface by default, although it is possible to connect this to an AXI3 system as necessary. Refer to the AMBA specification for details on how to connect AXI3 slaves to an AXI4 master.

Partial Store and Forward Using Packet Buffer DMA

When the DMA is configured to use SRAM based packet buffers, it can be programmed into a low latency mode, known as partial store and forward. This allows for a reduced latency as the full packet is not buffered before forwarding. Note that this option is only available when the device is configured for full duplex operation and when not using multi buffer frames.

This feature is enabled via the TX and RX partial store and forward programmable registers. When the transmit partial store and forward mode is activated, the transmitter will only begin to forward the packet to the MAC when there is enough packet data stored in the packet buffer. Likewise, when the receive partial store and forward mode is activated, the receiver will only begin to forward the packet to the external AHB or AXI slave when enough packet data is stored in the packet buffer.

The amount of packet data required to activate the forwarding process is programmable via watermark registers which are located at the same address as the partial store and forward enable bits. Note that the minimum operational value for the TX partial store and forward watermark is 20. There is no operational limit for the RX partial store and forward watermark. To reduce the bandwidth requirements of the receive buffer manager the receive buffer size can be increased above its default value of 128 bytes by writing to the DMA configuration register.

Enabling partial store and forward is a useful means to reduce latency, but there are performance implications. In essence, the packet buffer DMA will start behaving in a similar way to the internal FIFO DMA mode when partial store and forward is enabled. Information regarding this behavior is described later in this section of the document.

DMA Transactions

The DMA uses separate transmit and receive lists of buffer descriptors, with each descriptor describing a buffer area in memory. This can allow Ethernet packets to be broken up and scattered around the system memory (multi buffer operation), although one buffer per frame is also permitted.

The DMA controller performs four types of operation on the AMBA bus. When the GEM DMA is configured in internal FIFO mode, in order of priority these are:

1. *receive buffer manager write/read*
2. *transmit data DMA read*
3. *receive data DMA write*
4. *transmit buffer manager write/read*

When the GEM DMA is configured in AHB packet buffer mode, in order of priority these are:

1. *receive buffer manager write/read*
2. *transmit buffer manager write/read*
3. *receive data DMA write*
4. *transmit data DMA read*

When using AHB, the bus request signal `hbusreq` is asserted when the DMA block needs to perform any of these operations. The DMA block takes control of the bus when it sees both `hgrant` and `hready` asserted high.

When using AXI, all read operations are routed to the AXI read channel and all write operations to the write channel. Both read and write channels may operate simultaneously. Arbitration logic is required when multiple requests are active on the same channel (for example when the transmit DMA requests a transmit data read at the same time the receive DMA requests a receive descriptor read). In these cases, the receive DMA is granted the bus before the transmit DMA. However the vast majority of requests are either receive data writes or transmit data reads both of which can operate in parallel and can execute simultaneously.

Transfer size can be programmed to be 32, 64 or 128 bit words using the DMA bus width select bits in the network configuration register, and burst size may be programmed to single access or bursts of 4, 8, 16, or 256 words using the DMA Configuration Register.

Locked Accesses when using AHB and AHB Bus Ownership

The GEM DMA has been designed to support the loss of `hgrant` at any time during its transmit or receive AHB activities. However, there are certain implications tied to losing `hgrant` that the system integrator should be aware of:

During an AHB data burst, the GEM DMA will drive `hburst` appropriately. The loss of `hgrant` during a burst will cause all remaining AHB accesses of that burst to be cancelled thereby breaking the burst. Although the remaining accesses will be restarted when the grant is re-established, an AHB protocol error with respect to `hburst` will have occurred. If the system integrator requires the use of `hburst` and this behaviour is not acceptable, then the loss of grant during data bursts must be avoided.

When the GEM DMA is configured to be in the internal FIFO mode, or in the packet buffer partial store and forward mode there is a finite time that the DMA has to offload data to the AHB in the receive direction or fetch data from the AHB in the transmit direction. If `hgrant` is lost during DMA data transfers in a very frequent and/or prolonged manner, there is potential for internal fifo overflows or underflows causing occasional packet loss. The frequency of this scenario occurring can be minimized by increasing the size of the internal DMA buffers.

When the GEM DMA is configured to be in the packet buffer mode (and in full store and forward mode), all packets are fully buffered before being passed between the DMA and MAC. Therefore the loss of hgrant while in this mode has no effect on data integrity.

When the GEM DMA is configured in internal FIFO mode the `hlock` output of the DMA is asserted together with `hbusreq`. By connecting this `hlock` output to the AHB system arbiter, the grant to the DMA should never be removed during a locked transfer and consequently any potential issues discussed above associated with losing the grant will never occur. If the user wishes the grant to the DMA to be removed during certain operating conditions, the `hlock` output of the GEM should be left unconnected. When the GEM DMA is configured to be in the packet buffer mode, the `hlock` output of the DMA is tied low.

Receive DMA Buffers

Received frames, optionally including FCS, are written to receive buffers stored in AHB or AXI memory. The receive buffer depth is programmable in the range of 64 bytes to 16320 bytes. If received frames are being routed to different priority queues (via the packet inspection screeners – refer to section “Priority Queuing in the DMA”), it is possible to program different receive buffer depths for each queue. For queue 0, the receive buffer depth is programmed through the DMA Configuration register (offset 0x10). For the other queues, they are programmed in the independent queue configuration registers (starting from offset 0x4a0). The default is 128 bytes.

The start location for each receive buffer is stored in memory in a list of receive buffer descriptors at an address location pointed to by the receive buffer queue pointer. The base address for the receive buffer queue pointer is configured in software using the receive buffer queue base address register(s). For 64 bit addressing mode the msb buffer queue base address register can also be set and is valid for all receive buffer descriptors. With 64 bit addressing, there is a restriction that all the descriptors must be located within a region of memory that does not cross a 4GB region, in other words the upper 32 bits of the 64-bit address must be fixed. The actual 32 bits chosen for the upper bits are programmed in ms buffer queue base address register. This is only true of the descriptors and not the packet data which can be anywhere in the 64 bit address space.

The number of words in each buffer descriptor (BD) is dependent on the operating mode. Each BD word is defined as 32 bits.

The first two words (Word 0 and Word 1) are used for all BD modes.

In Extended Buffer Descriptor Modes (DMA configuration register bit 28 = 1), two BD words are added for 64 bit addressing mode and two BD words are added for timestamp capture. There are therefore either two, four or six BD words in each BD entry depending on operating mode, and every BD entry will have the same number of words. To summarize:-

- Every descriptor will be 64 bits wide when 64-bit addressing is disabled and the descriptor timestamp capture mode is disabled.
- Every descriptor will be 128 bits wide when 64-bit addressing is enabled and the descriptor timestamp capture mode is disabled.
- Every descriptor will be 128 bits wide when 64-bit addressing is disabled and the descriptor timestamp capture mode is enabled.
- Every descriptor will be 192 bits wide when 64-bit addressing is enabled and the descriptor timestamp capture mode is enabled.

The following description details the functionality of Word 0 and Word 1, but Word 2 must also be set when using 64 bit addressing mode. Each list entry consists of the same first two words. The first contains the start location of the receive buffer and the second the main receive status. If the length of a receive frame exceeds the DMA buffer length, the status word for the used buffer is written with zeroes except for the “start of frame” bit, which is always set for the first buffer in a frame. Bit zero of the address field is written to 1 to show the buffer has been used. The receive buffer manager then reads the location of the next receive buffer and fills that with the next part of the received frame data. Receive buffers are filled until the frame is complete and the final buffer descriptor status word contains the complete frame status. Refer to the receive buffer descriptor entry table for details of the receive buffer descriptor list.

When using receive descriptor timestamp capture, (DMA configuration register bit 28 = 1) bit 2 of Word 0 is used to indicate a valid timestamp has been captured in the BD. The use of bit 2 for this purpose also necessitates the Data Buffer being located on 64 bit address boundaries. Also note the timestamp can be considered status and will only be present for the final buffer descriptor of a frame.

Each receive buffer start location is a word address. The start of the first buffer in a frame can be offset by up to three bytes depending on the value written to bits 14 and 15 of the network configuration register. Note that when the `define gem_pbuf_rsc has been set then these bits are not used. For 64-bit datapaths, the start of the frame can be offset by up to a further four bytes if bit 2 of the DMA receive buffer start location in the buffer descriptor is set (This is applicable to the packet buffer DMA only – for the FIFO based DMA configured with a 64bit datapath, bit 2 of the buffer start location is ignored). If the start location of the buffer is offset the available length of the first buffer is reduced by the corresponding number of bytes.

Receive Buffer Descriptor Entry

Bit	Function
Word 0	
31:3	Address [31:3] of beginning of buffer.
2	Address [2] of beginning of buffer. Or In Extended Buffer Descriptor Mode (DMA configuration register[28] = 1), indicates a valid timestamp in the BD entry
1	Wrap - marks last descriptor in receive buffer descriptor list.
0	Ownership - needs to be zero for the GEM_GXL to write data to the receive buffer. The GEM_GXL sets this to 1 once it has successfully written a frame to memory. Software has to clear this bit before the buffer can be used again.
Word 1	
31	Global all ones broadcast address detected.
30	Multicast hash match.
29	Unicast hash match.
28	External address match. Note if the packet buffer mode and the number of configured specific address filters is greater than four in gem_gxl_defs.v then external address matching is not reported in this bit and instead it is set if there has been a match in the first eight specific address registers. Bit 27 is then

	used along with bits 26:25 to indicate which register matched.
27	Specific address register match found, bit 25 and bit 26 indicates which specific address register causes the match. See note for bit 28 above.
26:25	Specific address register match. Encoded as follows: 00 - Specific address register 1 match 01 - Specific address register 2 match 10 - Specific address register 3 match 11 - Specific address register 4 match If more than one specific address is matched only one is indicated with priority 4 down to 1.
24	This bit has a different meaning depending on whether RX checksum offloading is enabled. **With RX checksum offloading disabled:** (bit 24 clear in Network Configuration) Type ID register match found, bit 22 and bit 23 indicate which type ID register causes the match. **With RX checksum offloading enabled:** (bit 24 set in Network Configuration) 0 - the frame was not SNAP encoded and/or had a VLAN tag with the CFI bit set. 1 - the frame was SNAP encoded and had either no VLAN tag or a VLAN tag with the CFI bit not set.
23:22	This bit has a different meaning depending on whether RX checksum offloading is enabled. **With RX checksum offloading disabled:** (bit 24 clear in Network Configuration) Type ID register match. Encoded as follows: 00 - Type ID register 1 match 01 - Type ID register 2 match 10 - Type ID register 3 match 11 - Type ID register 4 match If more than one Type ID is matched only one is indicated with priority 4 down to 1. **With RX checksum offloading enabled:** (bit 24 set in Network Configuration) 00 - Neither the IP header checksum nor the TCP/UDP checksum was checked. 01 - The IP header checksum was checked and was correct. Neither the TCP nor UDP checksum was checked. 10 - Both the IP header and TCP checksum were checked and were correct. 11 - Both the IP header and UDP checksum were checked and were correct.
21	VLAN tag detected — type ID of 0x8100. For packets incorporating the stacked VLAN processing feature, this bit will be set if the second VLAN tag has a type ID of 0x8100
20	Priority tag detected — type ID of 0x8100 and null VLAN identifier. For packets incorporating the stacked VLAN processing feature, this bit will be set if the second VLAN tag has a type ID of 0x8100 and a null VLAN identifier.
19:17	When bit 15 (End of frame) and bit 21 (VLAN tag) are set, these bits represent the VLAN priority. When header/data splitting is enabled (via bit 5 of the DMA configuration register, offset 0x10) bit 17 indicates this descriptor is pointing to the last buffer of the header.

16	This bit has a different meaning depending on the state of bit 13 (report bad FCS in bit 16 of word 1 of the receive buffer descriptor) and bit 5 (header/data splitting) of the DMA Configuration register (offset 0x10). When header/data splitting is enabled and this buffer descriptor (BD) is not the last BD of the frame (as indicated in bit 15 of this BD), this bit will indicate that the BD is pointing to a data buffer containing header bytes. When this BD is the last BD of the frame (as indicated in bit 15 of this BD), and bit 13 of the DMA configuration register is set, this bit represents FCS/CRC error. When this BD is the last BD of the frame (as indicated in bit 15 of this BD), and bit 13 of the DMA configuration register is clear, and the received frame is VLAN tagged, this bit represents the Canonical format indicator (CFI).
15	End of frame - when set the buffer contains the end of a frame. If end of frame is not set, then the only valid status bit (unless header/data splitting is enabled) is start of frame (bit 14). If header/data splitting is enabled, then bits 16 and 17 are also valid status bits when this bit is not set.
14	Start of frame - when set the buffer contains the start of a frame. If both bits 15 and 14 are set, the buffer contains a whole frame.
13	This bit has a different meaning depending on whether jumbo frames and ignore FCS mode are enabled. If neither mode is enabled this bit will be zero. **With jumbo frame mode enabled:** (bit 3 set in Network Configuration Register) Additional bit for length of frame (bit[13]), that is concatenated with bits[12:0] **With ignore FCS mode enabled and jumbo frames disabled:** (bit 26 set in Network Configuration Register and bit 3 clear in Network Configuration Register) This indicates per frame FCS status as follows: 0 – Frame had good FCS 1 – Frame had bad FCS, but was copied to memory as ignore FCS enabled
12:0	When header/data splitting enabled (via bit 5 of the DMA configuration register, offset 0x10) and bit 17 is set (last buffer of header), these bits represent the length of the header in bytes. When bit 15 (End of frame) is set, these bits represent the length of the received frame which may or may not include FCS depending on whether FCS discard mode is enabled. **With FCS discard mode disabled:** (bit 17 clear in Network Configuration Register) Least significant 12-bits for length of frame **including** FCS. If jumbo frames are enabled, these 12-bits are concatenated with bit[13] of the descriptor above. **With FCS discard mode enabled:** (bit 17 set in Network Configuration Register) Least significant 12-bits for length of frame **excluding** FCS. If jumbo frames are enabled, these 12-bits are concatenated with bit[13] of the descriptor above.

When 64 bit Addressing mode is enabled, the following table identifies the added descriptor words:

Bit	Function
Word 2 (64 bit addressing)	

31:0	Upper 32 bit address of Data Buffer
	Word 3 (64 bit addressing)
31:0	Unused

When Descriptor Timestamp Capture mode is enabled, the following table identifies the added descriptor words:

Bit	**Function**
	Word 2 (32 bit addressing) or Word 4 (64 bit addressing)
31:30	Timestamp seconds[1:0] (See Note:1)
29:0	Timestamp nanosecs [29:0] (See Note:1)
	Word 3 (32 bit addressing) or Word 5 (64 bit addressing)
31:10	Unused
9:0	Timestamp seconds[11:2] (See Note:1) (prior to release 1p08f1 this was [5:2])
	Note1: The timestamp is mode is controlled using the rx_bd_control_register. The RX Descriptor Timestamp Insertion mode bits are defined as, 00: TS insertion disable, 01: TS inserted for PTP Event Frames only, 10: TS inserted for All PTP Frames only, 11: TS insertion for All Frames. The timestamp bits are written back to the last buffer descriptor of a frame only.

To receive frames, the receive buffer descriptors must be initialized by writing an appropriate address to bits 31:2 (or 31:3 for timestamp capture mode) in the first word of each list entry. Bit 0 must be written with zero. Bit 1 is the wrap bit and indicates the last entry in the buffer descriptor list.

The start location of the receive buffer descriptor list must be written with the receive buffer queue base address before reception is enabled (receive enable in the network control register). Once reception is enabled, any writes to the receive buffer queue base address register are ignored. When read, it will return the current pointer position in the descriptor list, though this is only valid and stable when receive is disabled.

If the filter block indicates that a frame should be copied to memory, the receive data DMA operation starts writing data into the receive buffer. If an error occurs, the buffer is recovered.

An internal counter within the GEM_GXL represents the receive buffer queue pointer and it is not visible through the CPU interface. The receive buffer queue pointer increments by two, four or six words after each buffer has been used, depending on the descriptor size. It re-initializes to the receive buffer queue base address if any descriptor has its wrap bit set. Note that this operation is different from Cadence's Ethernet MAC 10/100 (MACB), which will also wrap after 1024 buffers have been used.

As receive buffers are used, the receive buffer manager sets bit zero of the first word of the descriptor to logic one indicating the buffer has been used.

Software should search through the "used" bits in the buffer descriptors to find out how many frames have been received, checking the start of frame and end of frame bits.

When the DMA is configured for internal FIFO mode, received frames are written out to the AHB buffers as soon as the frame is matched by the filtering logic even though there may still be more data to be received. Similarly when the DMA is configured in the packet buffer partial store and forward mode, received frames are written out to the AHB/AXI buffers as soon as enough frame data exists in the packet buffer. For both cases, this may mean several full buffers are used before some error conditions can be detected. If a receive error is detected the receive buffer currently being written will be recovered. Previous buffers will not be recovered. As an example, when receiving frames with CRC errors or excessive length, it is possible that a frame fragment might be stored in a sequence of receive buffers. Software can detect this by looking for start of frame bit set in a buffer following a buffer with no end of frame bit set.

For a properly working 10/100/1000 Ethernet system there should be no excessive length frames or frames greater than 128 bytes with CRC errors. Collision fragments will be less than 128 bytes long, therefore it will be a rare occurrence to find a frame fragment in a receive buffer, when using the default value of 128 bytes for the receive buffers size.

When in packet buffer full store and forward mode only good received frames are written out of the DMA, so no fragments will exist in the buffers due to MAC receiver errors. There is still the possibility of fragments due to DMA errors, for example used bit read on the second buffer of a multi-buffer frame.

If bit zero of the receive buffer descriptor is already set when the receive buffer manager reads the location of the receive buffer, then the buffer has been already used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the "buffer not available" bit in the receive status register is set and an interrupt triggered. The receive resource error statistics register is also incremented.

When the DMA is configured in the packet buffer full store and forward mode, the user can optionally select whether received frames should be automatically discarded when no buffer resource is available. This feature is selected via bit 24 of the DMA Configuration register (by default, the received frames are not automatically discarded). If this feature is off, then received packets will remain to be stored in the SRAM based packet buffer until AHB or AXI buffer resource next becomes available. This may lead to an eventual packet buffer overflow if packets continue to be received when bit zero (used bit) of the receive buffer descriptor remains set. Note that after a used bit has been read, the receive buffer manager will reread the location of the receive buffer descriptor every time a new packet is received. When the DMA is not configured in the packet buffer full store and forward mode and a used bit is read, the frame currently being received will be automatically discarded.

When the DMA is configured in the packet buffer full store and forward mode, a receive overrun condition occurs when the receive SRAM based packet buffer is full, or because an AMBA error occurred (via `hresp or bresp`). In all other modes, a receive overrun condition occurs when either the AHB/AXI bus was not granted quickly enough, or because an AMBA AHB or AXI error occurred, or because a new frame has been detected by the receive block when the status update or write back for the previous frame has not yet finished. For a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame that is received whose address is recognized reuses the buffer.

When the DMA is configured for packet buffer mode, a write to bit 18 of the network control register will force a packet from the external SRAM based receive packet buffer to be flushed. This feature is only acted upon when the RX DMA is not currently writing packet data out to memory– i.e. it is in an IDLE state. If the RX DMA is active, a write to this bit is ignored.

When the DMA is configured for packet buffer mode, the upper bits of the data buffer address stored in bits 31:2 in the first word of each list entry can be dynamically altered in real-time without physically changing the external memory holding the list entry. This feature is useful if the user needs to select the destination based on CPU usage or other flow control hardware. It is achieved using a MUX structure whereby the user can define whether the upper 4 bits of the 32-bit data-buffer AHB or AXI address should come from the descriptor list entry or from a programmable register. Refer to the “Receive Data Buffer Address Mask” programmable register for further details. Note that any changes to this register will be ignored while the DMA is currently processing a receive packet. It will only affect the next full packet to be written to AHB or AXI memory.

Transmit Buffers

Frames to transmit are stored in one or more transmit buffers. Transmit frames can be between 1 and 16383 bytes long, so it is possible to transmit frames longer than the maximum length specified in the IEEE 802.3 standard. It should be noted that zero length buffers are allowed and that the maximum number of buffers permitted for each transmit frame is 128.
The start location for each transmit buffer is stored in AHB or AXI memory in a list of transmit buffer descriptors at a location pointed to by the transmit buffer queue pointer. The base address for this queue pointer is set in software using the transmit buffer queue base address register(s). For 64 bit addressing mode the ms buffer queue base address register can also be set and is valid for all transmit buffer descriptors With 64 bit addressing, there is a restriction that all the descriptors must be located within a region of memory that does not cross a 4GB region, in other words the upper 32 bits of the 64-bit address must be fixed. The actual 32 bits chosen for the upper bits are programmed in ms buffer queue base address register. This is only true of the descriptors and not the packet data which can be anywhere in the 64 bit address space.

The number of words in each BD is dependent on the operating mode.

The first two words (Word 0 and Word 1) are used for all BD modes.

In Extended Buffer Descriptor Modes, two BD words are added for 64 bit addressing mode and two BD words are added for timestamp capture. There are therefore either two, four or six BD words in each BD entry depending on operating mode, and every BD entry will have the same number of words. To summarize:-

- Every descriptor will be 64 bits wide when 64-bit addressing is disabled and the descriptor timestamp capture mode are disabled.
- Every descriptor will be 128 bits wide when 64-bit addressing is enabled and the descriptor timestamp capture mode are disabled.
- Every descriptor will be 128 bits wide when 64-bit addressing is disabled and the descriptor timestamp capture mode are enabled.
- Every descriptor will be 192 bits wide when 64-bit addressing is enabled and the descriptor timestamp capture mode are enabled.

The following description details the functionality of Word 0 and Word 1, but Word 2 must also be set when using 64 bit addressing mode.

Each list entry consists of the same first two words. The first is the byte address of the transmit buffer and the second containing the transmit control and status. For the packet buffer DMA, the start location for each AHB or AXI buffer is a byte address, the bottom bits of the address being used to offset the start of the data from the data-word boundary(i.e. bits 2,1 and 0 are used to offset the address for 64-bit datapaths). For the FIFO based DMA configured with a 32-bit datapath the address of the buffer is also a byte address. For bus widths of 64 or 128-bits however, the address of the buffer must be aligned to the correct 64-bit or 128-bit boundary, plus an offset of less than 4 bytes (Note this alignment restriction in FIFO based DMA mode only should be sufficient for applications as the main purpose is to allow alignment of the encapsulated IP packet - Given the 14 bytes of MAC encapsulation, an offset of 2 will always align the IP header to a 128-bit boundary)

Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, pad will also be automatically generated to take frames to a minimum length of 64 bytes. When CRC is not automatically generated (as defined in word 1 of the transmit buffer descriptor or through the control bus of the external FIFO interface), the frame is assumed to be at least 64 bytes long and pad is not generated.

An entry in the transmit buffer descriptor list is described in table *Transmit Buffer Descriptor Entry* – Non-LSO Frame

To transmit frames, the buffer descriptors must be initialized by writing an appropriate byte address to bits 31:0 in the first word of each descriptor list entry.

The second word of the transmit buffer descriptor is initialized with control information that indicates the length of the frame, whether or not the MAC is to append CRC and whether the buffer is the last buffer in the frame.

After transmission the status bits are written back to the second word of the first buffer descriptor along with the used bit. Bit 31 is the used bit which must be zero when the control word is read if transmission is to take place. It is written to one once the frame has been transmitted. Bits[29:20] indicate various transmit error conditions. Bit 23 indicates a valid timestamp has been captured in the BD. Bit 30 is the wrap bit which can be set for any buffer within a frame. If no wrap bit is encountered the queue pointer continues to increment. This operation is different from Ethernet MAC 10/100 (MACB) from Cadence, which will wrap after 1024 buffers have been used.

The transmit buffer queue base address register can only be updated whilst transmission is disabled or halted; otherwise any attempted write will be ignored. When transmission is halted the transmit buffer queue pointer will maintain its value. Therefore when transmission is restarted the next descriptor read from the queue will be from immediately after the last successfully transmitted frame. Whilst transmit is disabled (bit 3 of the network control set low), the transmit buffer queue pointer resets to point to the address indicated by the transmit buffer queue base address register. Note that disabling receive does not have the same effect on the receive buffer queue pointer.

Once the transmit queue is initialised, transmit is activated either by writing to the transmit start bit (bit 9) of the network control register, or in hardware by toggling the `trigger_dma_tx_start` input. Transmit is halted when a buffer descriptor with its used bit set is read, a transmit error occurs, or by writing to the transmit halt bit of the network control register. Transmission is suspended if a pause frame is received while the pause enable bit is set in the network configuration register. Rewriting the start bit while transmission is active is allowed. This is implemented with a `tx_go` variable which is readable in the transmit status register at bit location 3. The `tx_go` variable is reset when:

- Transmit is disabled.
- A buffer descriptor with its ownership bit set is read.
- Bit 10, `tx_halt`, of the network control register is written.
- There is a transmit error such as too many retries, late collision (gigabit mode only) or a transmit under run.

To set `tx_go` write to bit 9, `tx_start`, of the network control register. Transmit halt does not take effect until any ongoing transmit finishes.

If the DMA is configured for internal FIFO mode and a collision occurs during transmission of a multi-buffer frame transmission will automatically restart from the first buffer of the frame. For packet buffer mode, the entire contents of the frame are read into the transmit packet buffer memory, so the retry attempt will be replayed directly from the packet buffer memory rather than having to re-fetch through the AHB or AXI.

If the transmit buffer list is incorrectly set up in such a way that a used bit is read mid-way through a multi buffer frame, transmission will stop. If cut-through is in operation and the MAC has actually starting transmitting the frame which has its used bit set, the MAC treats it as a transmit error, and asserts `tx_er` truncates the frame and corrupts the FCS.

Transmit Buffer Descriptor Entry – Non-LSO Frame

Bit	Function
Word 0	
31:0	Byte address of buffer.
Word 1	
31	Used – must be zero for the GEM_GXL to read data to the transmit buffer. The GEM_GXL sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap – marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29	Retry limit exceeded, transmit error detected
28	Transmit under run – occurs when the start of packet data has been written into the FIFO and either `hresp` is not OK, or the transmit data could not be fetched in time, or when buffers are exhausted. This is not set when the DMA is configured for packet buffer mode.
27	Transmit frame corruption due to AHB or AXI error – set if an error occurs whilst midway through reading transmit frame from the AHB/AXI, including HRESP or RRESP/BRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and `tx_er` asserted). Also set in AHB (not AXI) DMA packet buffer mode if single frame is too large for configured packet buffer memory size.
26	Late collision, transmit error detected. Late collisions only force this status bit to be set in gigabit mode.
25:24	Reserved.
23	For Extended Buffer Descriptor Mode this bit Indicates a timestamp has been captured in the BD. Otherwise Reserved.
22:20	Transmit IP/TCP/UDP checksum generation offload errors: 000 No Error 001 The Packet was identified as a VLAN type, but the header was not fully complete, or had an error in it 010 The Packet was identified as a SNAP type, but the header was not fully complete, or had an error in it 011 The Packet was not of an IP type, or the IP packet was invalidly short, or the IP was not of type IPv4/IPv6 100 The Packet was not identified as VLAN, SNAP or IP 101 Non supported packet fragmentation occurred. For IPv4 packets, the IP checksum was generated and inserted 110 Packet type detected was not TCP or UDP. TCP/UDP checksum was therefore not generated. For IPv4 packets, the IP checksum was generated and inserted 111 A premature end of packet was detected and the TCP/UDP checksum could not be generated
19:17	Reserved. Must be set to 3’b000 to disable TSO and UFO

16	No CRC to be appended by MAC. When set this implies that the data in the buffers already contains a valid CRC and hence no CRC or padding is to be appended to the current frame by the MAC. This control bit must be set for the first buffer in a frame and will be ignored for the subsequent buffers of a frame. This operation is different from Cadence's Ethernet MAC 10/100 (Enhanced), which reads the no CRC bit from the final buffer descriptor in the frame. Note that this bit must be clear when using the transmit IP/TCP/UDP checksum generation offload, otherwise checksum generation and substitution will not occur. Note. This bit must also be cleared when TX Partial Store and Forward mode is active.
15	Last buffer, when set this bit will indicate the last buffer in the current frame has been reached.
14	Reserved.
13:0	Length of buffer.

Transmit Buffer Descriptor Entry – TSO Header Buffer

Bit	Function
	Word 0
31:0	Byte address of buffer.
	Word 1
31	Used – must be zero for the GEM_GXL to read data to the transmit buffer. The GEM_GXL sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap – marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29	Retry limit exceeded, transmit error detected
28	Transmit under run – always 0 for TSO
27	Transmit frame corruption due to AHB or AXI error – set if an error occurs whilst midway through reading transmit frame from the AHB/AXI, including HRESP or RRESP/BRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and `tx_er` asserted). Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size.
26	Late collision, transmit error detected. Late collisions only force this status bit to be set in gigabit mode.
25:24	TCP Stream Identifier. Used to select the hardware counter that is used for TCP sequence number generation.

23	For Extended Buffer Descriptor Mode this bit Indicates a timestamp has been captured in the BD. Otherwise Reserved.
22:20	Transmit IP/TCP/UDP checksum generation offload errors: 000 No Error 001 The Packet was identified as a VLAN type, but the header was not fully complete, or had an error in it 010 The Packet was identified as a SNAP type, but the header was not fully complete, or had an error in it 011 The Packet was not of an IP type, or the IP packet was invalidly short, or the IP was not of type IPv4/IPv6 100 The Packet was not identified as VLAN, SNAP or IP 101 Non supported packet fragmentation occurred. For IPv4 packets, the IP checksum was generated and inserted 110 Packet type detected was not TCP or UDP. TCP/UDP checksum was therefore not generated. For IPv4 packets, the IP checksum was generated and inserted 111 A premature end of packet was detected and the TCP/UDP checksum could not be generated
19	TCP Sequence Number Source Select 0 – Use sequence number value from the header buffer for the first small TCP frame and hardware generated sequence number values for subsequent small TCP frames 1 – Use hardware generated sequence number values for all small TCP frames
18:17	LSO Control : Set to 2'b10 or 2'b11 to enable TSO
16	No CRC to be appended by MAC. Must be clear for TSO operation
15	Last buffer : Must be clear as TSO requires at least one payload buffer
14	Reserved.
13:0	Length of buffer.

Transmit Buffer Descriptor Entry – TSO Payload Buffer

Bit	Function
	Word 0
31:0	Byte address of buffer.
	Word 1
31	Used – must be zero for the GEM_GXL to read data to the transmit buffer. The GEM_GXL sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap – marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29:16	TCP Maximum Segment Size value in bytes. TSO will use a default value of

	536 bytes if the programmed value is zero.
15	Last buffer, when set this bit will indicate the last buffer in the current frame has been reached.
14	Reserved.
13:0	Length of buffer.

Transmit Buffer Descriptor Entry – UFO Header Buffer

Bit	Function
	Word 0
31:0	Byte address of buffer.
	Word 1
31	Used – must be zero for the GEM_GXL to read data to the transmit buffer. The GEM_GXL sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap – marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29	Retry limit exceeded, transmit error detected
28	Transmit under run – always 0 for UFO
27	Transmit frame corruption due to AHB or AXI error – set if an error occurs whilst midway through reading transmit frame from the AHB/AXI, including HRESP or RRESP/BRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and `tx_er` asserted). Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size.
26	Late collision, transmit error detected. Late collisions only force this status bit to be set in gigabit mode.
25:24	Reserved
23	For Extended Buffer Descriptor Mode this bit Indicates a timestamp has been captured in the BD. Otherwise Reserved.
22:20	Transmit IP/TCP/UDP checksum generation offload errors: 000 No Error 001 The Packet was identified as a VLAN type, but the header was not fully complete, or had an error in it 010 The Packet was identified as a SNAP type, but the header was not fully complete, or had an error in it 011 The Packet was not of an IP type, or the IP packet was invalidly short, or the IP was not of type IPv4/IPv6 100 The Packet was not identified as VLAN, SNAP or IP

	101 Non supported packet fragmentation occurred. For IPv4 packets, the IP checksum was generated and inserted 110 Packet type detected was not TCP or UDP. TCP/UDP checksum was therefore not generated. For IPv4 packets, the IP checksum was generated and inserted 111 A premature end of packet was detected and the TCP/UDP checksum could not be generated
19	Reserved
18:17	LSO Control : Set to 2’b01 to enable UFO
16	No CRC to be appended by MAC. Must be clear for UFO operation
15	Last buffer : Must be clear as TSO requires at least one payload buffer
14	Reserved.
13:0	Length of buffer.

Transmit Buffer Descriptor Entry – UFO Payload Buffer

Bit	Function
	Word 0
31:0	Byte address of buffer.
	Word 1
31	Used – must be zero for the GEM_GXL to read data to the transmit buffer. The GEM_GXL sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap – marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29:16	Maximum Ethernet Frame Size value in bytes (including FCS). UFO will use a default value of 1518 bytes if the programmed value is zero.
15	Last buffer, when set this bit will indicate the last buffer in the current frame has been reached.
14	Reserved.
13:0	Length of buffer.

When 64 bit Addressing mode is enabled, the following table identifies the added descriptor words:

Bit	Function

Word 2 (64 bit addressing)	
31:0	Upper 32 bit address of Data Buffer
Word 3 (64 bit addressing)	
31:0	Unused

When Descriptor Timestamp Capture mode is enabled, the following table identifies the added descriptor words:

Bit	Function
Word 2 (32 bit addressing) or Word 4 (64 bit addressing)	
31:30	Timestamp seconds[1:0] (See Note:1)
29:0	Timestamp nanosecs [29:0] (See Note:1)
Word 3 (32 bit addressing) or Word 5 (64 bit addressing)	
31:10	Unused
9:0	Timestamp seconds[11:2] (See Note:1) (prior to release 1p08f1 this was [5:2])
	Note1: The timestamp is mode is controlled using the tx_bd_control_register. The TX Descriptor Timestamp Insertion mode bits are defined as, 00: TS insertion disable, 01: TS inserted for PTP Event Frames only, 10: TS inserted for All PTP Frames only, 11: TS insertion for All Frames. After transmission the timestamp bits are written back only to the first buffer descriptor.

DMA Bursting

The AXI DMA will always use INCR type accesses. The AHB DMA will always use SINGLE, or INCR type AHB accesses. When performing data transfers, the burst length used can be programmed using bits [4:0] of the DMA Configuration Register. Either single or fixed length incrementing bursts up to a maximum of 256 (AXI4) or 16 (AHB) are used as appropriate.

When there is sufficient space and enough data to be transferred, the programmed fixed length bursts will be used. If there is not enough data or space available, for example when at the end of a buffer or packet, bursts of sizes less than the programmed burst length value will be issued. The same is true at 1024 byte boundaries for AHB or 4096 byte boundaries for AXI, so that the relevant boundaries are not burst over as per AMBA requirements. Where the number of accesses remaining is less than 4 for AHB, single accesses will be used.

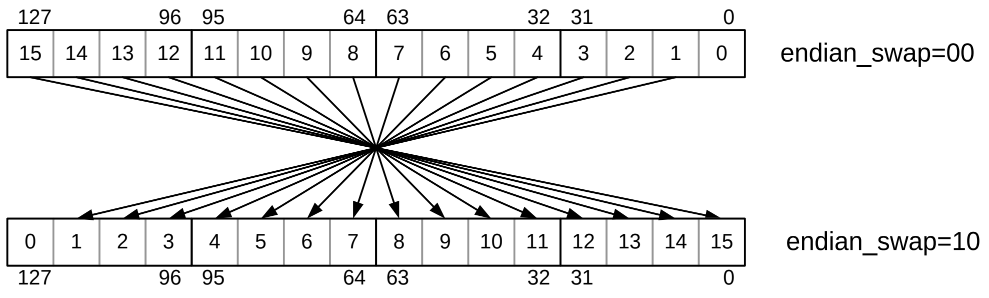
When the DMA is configured for packet buffer mode, an option to force the GEM DMA to pad the remaining bursts at the end of a buffer or EOP to the programmed burst length value is available via bits 26 and 25 of the DMA Configuration Register. Bit 26 will control the TX and bit 25 the RX. For RX, the data to burst is padded with ‘0’s up to the burst boundary defined by burst length. For TX, the extra data that is read is ignored by the DMA. This feature has been included for performance reasons when external AHB or AXI slaves that are being accessed by the GEM perform better when accessed using fixed max length bursts. Note enabling this feature will not break the AHB 1Kbyte rule, or the AXI 4KB rule and the enable bits are ignored if the programmed burst length is 256.

The DMA will not terminate fixed length bursts early, unless an AHB HRESP error condition is detected (note AXI response errors will never cause a burst to be terminated early) or if receive/transmit operation is disabled by software by writing to the Network Control register.

DMA Endianism

The default configuration of the DMA is to use little endian format. In order to interface to different CPU systems with different endianism requirements, the GEM DMA may be programmed to swap the endianism using bits 6 and 7 of the DMA Configuration Register. Bit 6 controls the endianism of the management operations and bit 7 controls the endianism of the data operations.

When either of these modes are enabled, the order of the bytes on the AHB or AXI are swapped for either or both of the buffer management and data operations, as illustrated in the diagram below for a 128-bit system for data operations (bit 7 set). The same swapping applies for management operations if bit 6 is set.



DMA Transmit and Receive FIFOs

This section is not relevant if the GEM_GXL is configured to use SRAM based packet buffer DMA mode.

The DMA can be configured to use a very low latency mode by not including the packet buffer. In this mode separate transmit and receive FIFOs are including within the DMA which are implemented using synthesisable flip-flop arrays. The FIFOs are used to smooth out dataflow delays and latencies on the AHB fabric to ensure consistent service for the MAC pipelines and prevent underflows and overflows. The FIFOs are also used to transition data over the clock boundary between the DMA and the MAC `rx_clk` and `tx_clk`.

The transmitter and receiver FIFO depths are parameterizable. It is important to choose FIFO depths that are appropriate for the system clock speed, memory latency and network speed. In the default implementation of the GEM, the receive and transmit FIFO depths are both set to ten locations each of 128 bits, giving a total of 160 bytes of storage per FIFO.

Data is typically transferred into and out of the FIFOs in bursts of the programmed burst length in the DMA configuration register.

For receive, an AHB bus request is asserted when the FIFO contains greater than or equal to the programmed AHB burst length. When the end of frame is reached, single word transfers may be used to complete the frame transfer if there is less data left in the frame to transfer than the programmed burst length. If the frame does not finish on the word boundary — depending on `dma_bus_width`, the frame length and the initial byte offset — the last word of the last transfer will contain redundant bytes, written as logic zero.

For transmit, an AHB bus request is generated whenever there is space for greater than or equal to the programmed AHB burst length in the transmit FIFO. This sequence is followed for every transfer in the buffer except the last. Once the last data transfer of a buffer is reached, the final burst may consist of single transfers if there is less data left in the frame to transfer than the programmed burst size. The last word of the last transfer contains redundant bytes if the frame does not finish on the word boundary. This depends upon `dma_bus_width` and the frame length.

Transmission of the frame will not begin until as much as the frame as possible has been read into the transmit FIFO. This is deemed to be when there is less space left in the FIFO than the programmed burst length, or if the end of the frame has already been written into the FIFO. Once either of these events occurs a trigger is sent to the MAC transmitter, which will then begin reading out the frame from the FIFO. This way there is more of a buffer built up to allow for any delays in the AHB which may otherwise cause an underflow of the FIFO.

The system designer should consider data bus width, transmit/receive clock rates, AMBA DMA bus latency and the chosen AMBA AHB clock rate. These parameters will determine the FIFO depth.

The default FIFO depth of 10 means that the bus latency must be less than the time it takes to load the FIFO and transmit or receive six words — 24 bytes in 32 bit AHB mode — of data. At 1000 Mbps, it takes 192 ns to transmit or receive 24 bytes of data. In addition six `hclk` cycles should be allowed for data to be loaded from the bus and to propagate through the FIFOs. For an 80 MHz `hclk` this takes 75 ns, therefore making the bus latency requirement approximately 120 ns.

Transmit buffers can be any length between 1 and 16383 bytes. For short length buffers, adequate bandwidth must be allowed for the buffer management overhead. It is recommended to perform simulations to ensure the system has adequate bandwidth if transmit frames are likely to contain a number of very short buffers. If the DMA block cannot access transmit data from memory in time, the “transmit under run” status will be reported and an interrupt asserted.

Note that the FIFO depths must be chosen to allow for the desired burst length. For instance if a 16-beat burst is desirable then the depths of both the receiver and transmitter FIFOs must be at least 16 words deep. The DMA Configuration register automatically prevents programming of a burst length that is greater than the FIFOs depth, by masking out the appropriate upper bits of the burst length field in the register.

AXI operation is not supported when the DMA is configured in this mode.

DMA Packet Buffer

This section is only relevant if the GEM is configured to use SRAM based packet buffer DMA mode.

The DMA can be configured to use packet buffers for both transmit and receive paths. In this mode, the user has the option of an AXI4 master interface or an AHB master interface.

This mode allows multiple packets to be buffered in both transmit and receive directions and allows the DMA to withstand variable levels of access latencies on the AHB or AXI fabric. This mode offers the most efficient use of the system bandwidth.

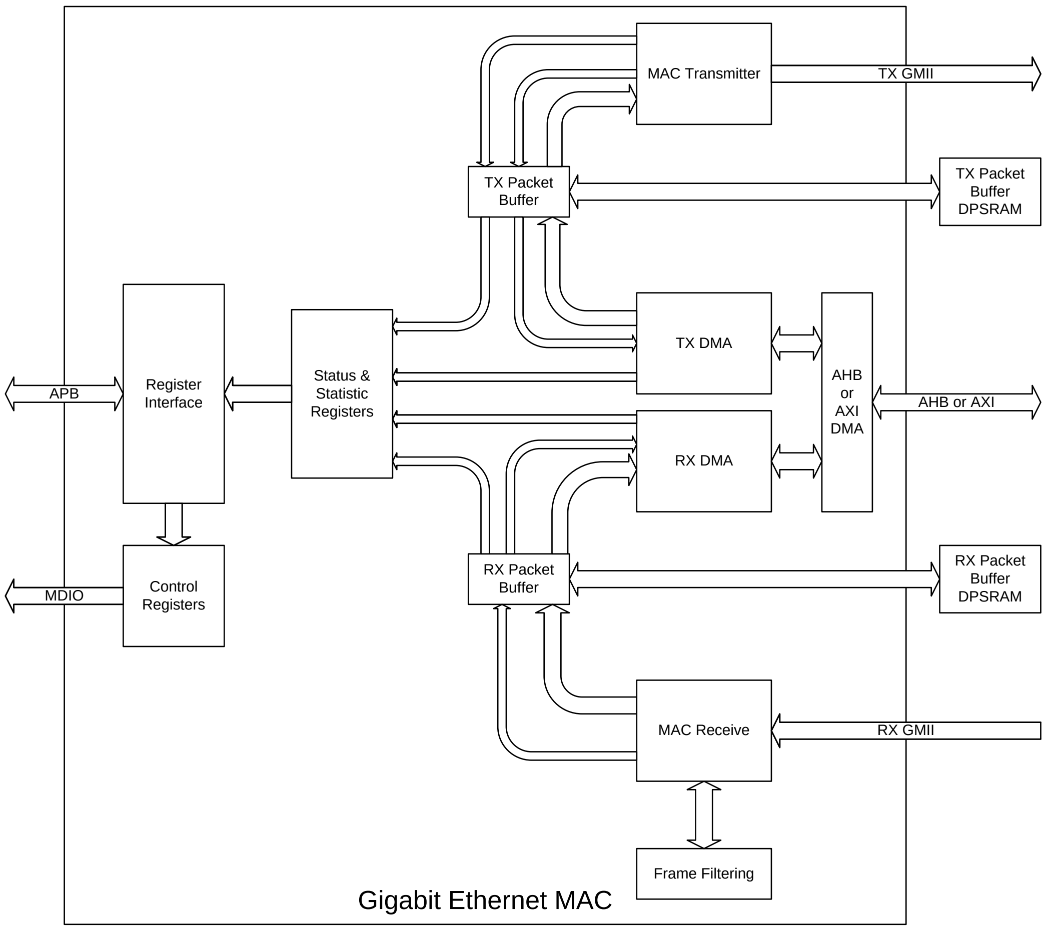
As described earlier, when the DMA is configured to use packet buffers, it can be programmed into a low latency mode known as partial store and forward. When this is enabled the packet buffer DMA will functionally behave in a similar way to the internal FIFO DMA mode. Further details of this mode have already been given earlier in the document and are not repeated here.

When the DMA is programmed in full store and forward mode full packets are buffered providing the opportunity to:-

- Discard packets that are received with errors before they are partially written out of the DMA thus saving system bus bandwidth and driver processing overhead,

- Retry collided transmit frames from the buffer, thus saving AHB or AXI bus bandwidth,
- Implement transmit IP/TCP/UDP checksum generation offload.

The following illustrates the structure of the GEM_GXL data paths when the packet buffers are included.



External Memories:

Two additional memories are required to be added externally to the GEM_GXL to provide the packet buffering. These must be connected to the transmitter and receiver packet buffer interfaces at the top level. The external memory type can either be dual port or single port.

The packet buffer implementation works with 32-bit, 64-bit or 128-bit AMBA databus widths.

In the transmit direction, the DMA will continue to fetch packet data up to a limit of 256 packets, or until either the external buffer is full. The size of the external buffer may be programmed to use either all or half the address range as specified in the configuration file. In the default configuration, the buffer address width is set to 10 bits, although the actual size required by the user can be set via the design configuration file. In 32-bit mode, a setting of 10 bits relates to a maximum useable size of 4 Kbytes. Similarly, in 64-bit mode it relates to a size of 8 Kbytes.

In the receive direction, if the external buffer becomes full, then an overflow will occur. An overflow will also occur if the limit of 256 packets is breached. The size of the external buffer may be programmed to use either all, half, quarter or an eighth of the address range as specified in the configuration file. In the default configuration, the buffer address width is set to 11 bits. In 32-bit mode, this relates to a maximum external buffer size of 8 Kbytes. Similarly, in 64-bit mode it relates to a size of 16 Kbytes. Again, these sizes may be scaled during configuration.

Determining RAM Size

To guarantee maximum bandwidth in all modes of operation, the general rule for the RAM is a size equal to twice the anticipated maximum frame size. Having a RAM that is twice the maximum frame size allows one frame to be forwarded while the next frame is being written. For example, in the transmit direction one frame will be transmitted through the MAC while another frame is being read from the host AMBA interface. This configuration ensures maximum throughput when maximum sized frames are transmitted / received consecutively, as both halves of the DMA can operate concurrently.

For transmit, each priority queue has an independent reserved area of RAM, and the above rule should be met on each queue’s RAM segment – I.e. When priority queuing mode and full store and forward mode is enabled, every configured region of the attached memory attributed to a priority queue must be greater than 2X the maximum transmitted frame length. For receive, all queues share the single RAM space and the above rule need only be applied to the full RAM. Using a 4 queue configuration as an example and a maximum sized frame of 15kB then to guarantee maximum performance in all modes of operation, the following configuration should be used:

TX RAM Sizes	128kB	Queue 3	32kB
		Queue 2	32kB
		Queue 1	32kB
		Queue 0	32kB
RX RAM Sizes	32kB		

In the above table the transmit path is configured to have 4 segments, with each segment having an equal size.

Reducing RAM Size

The RAM sizes described above are cautious and are not always necessary. If maximum sized frames are not transmitted consecutively, then a smaller RAM size is permitted. In full store and forward mode, the minimum RAM size for correct functional operation must be greater than the max frame length. Also, if the core is configured in full duplex operation, the RAM size on the transmit side can be reduced as detailed below.

Reducing RAM Size – Transmit Operation

The transmit DMA design is efficient and recovers memory as frames are transmitted through the MAC. After each byte is transmitted the memory area used by this byte will be recovered. By recovering memory locations as frames are transmitted then space is available for the next frame and this frame can be read over the AMBA bus. If the maximum frame size is less than 80% of the allocated transmit RAM size reserved per queue, and the average AXI or AHB bandwidth is significantly higher than the required line rate, then it is unlikely that there will be a performance impact. The 20% overhead allows the DMA to maintain overall line rate. Note. This operation does not apply in half duplex mode as the frame may need retransmitted owing to a collision and the memory used by the transmit frame is only cleared after the frame has fully transferred through the MAC. The memory size for transmitting a maximum frame length of for example 10KB could be transmitted and received at line rate with the following SRAM sizes:

TX RAM Sizes	64kB	Queue 3	16kB
		Queue 2	16kB
		Queue 1	16kB
		Queue 0	16kB
RX RAM Sizes	32kB		

Reducing RAM Size – Partial Store and Forward Mode

Partial store and forward mode allows frames that are larger than the RAM size to transfer through the design. If under normal circumstances the maximum sized frame is 2kB, but frames larger than this size may occasionally occur then partial store and forward mode could be activated with a threshold of 2kB. In this scenario the buffer sizes could be reduced to the following:

TX RAM Sizes	16kB	Queue 3	4kB
		Queue 2	4kB
		Queue 1	4kB
		Queue 0	4kB
RX RAM Sizes	4kB		

When 2kB of data is has been written to memory then it will be forwarded, regardless of whether or not the full frame has been written. The memory used by this initial 2kB will be recovered as the frame is forwarded, which allows the remainder of the frame to be written. With partial store and forward mode there is the risk of underflow (transmit path) or overflow (receive path) if the AMBA interface does not keep up with the frame rate. This risk is however low if the AMBA rate is faster than the MAC rate – in most systems it is typically faster and this underflow/overflow risk is low. Note if partial store and forward mode is to be used, then the define 'pbuf_cutthru' should be set in the configuration file.

Dual Port External Memory:

Note that although these memories must have 2 ports, each port has only a single access direction. I.e. port A of the TX DPRAM is WRITE only, while port B is READ only.

The external DPRAM modules that must allow fully independent read and write interfaces using different clock domains and independent addressing. The packet buffer implementation will ensure that the same memory address is not accessed simultaneously by both the read and write ports. This is effectively an underflow condition of the buffer which is not possible.

The connections to the DPRAM modules are as follows:-

- The write port of the transmitter DPRAM must be connected to the `txdpram_*a` signals, all of which are AMBA `hclk (or aclk for AXI)` timed and hence this port must be clocked by `hclk/aclk`.

- The read port of the transmitter DPRAM must be connected to the `txdpram_*b` signals, all of which are MAC tx_clk timed and hence this port must be clocked by tx_clk.
- The write port of the receiver DPRAM must be connected to the `rxdpram_*a` signals, all of which are MAC `rx_clk` timed and hence this port must be clocked by `rx_clk`.
- The read port of the receiver DPRAM must be connected to the `rxdpram_*b` signals, all of which are AMBA hclk timed and hence this port must be clocked by `hclk/aclk`.
-

Single Port External Memory:

GEM_GXL can be configured to optionally use an external single port memory. For this mode of operation all memory signals are synchronous to aclk/hclk and both the receive and transmit memory must be therefore be connected to aclk/hclk.

Dual port external memory has no frequency limitations on aclk/hclk and these clocks can therefore run faster or lower than the MAC clocks (rx_clk and tx_clk). Single port external memory operation however has some frequency limitations, where the aclk/hclk frequency must be faster than the MAC data rate. The following table details recommended minimum operating frequencies for all possible configurations and MAC speeds:

DMA Bus Width	**MAC Rate**	**Minimum AHB/AXI Frequency**
32	1Gbps	100MHz
32	100Mbps	10MHz
32	10Mbps	1MHz
64	1Gbps	50MHz
64	100Mbps	5MHz
64	10Mbps	0.5MHz
128	Not a supported configuration with a single port external memory configuration	Not applicable

GEM does not have automatic built-in support for powering down external memory when using power saving modes of operation. GEM however offers general purpose IO that can instead be used to power down or activate stand by modes of operation when necessary.

Transmit Packet Buffer:

The transmitter packet buffer will continue attempting to fetch frame data from the AHB or AXI system memory until the packet buffer itself is full, at which point it will attempt to maintain its full level.

To accommodate the status and statistics associated with each frame, 3 words (or 2 if the GEM_GXL is configured in 64 bit datapath mode) per packet are reserved at the end of the packet data. If the packet was bad and requires to be dropped, the status and statistics are the only information held on that packet. Storing the status in the DPRAM is required in order to decouple the DMA interface of the buffer from the MAC interface, to update the MAC status/stats and to generate interrupts in the order in which the packets that they represent were fetched from the external memory.

The transmit packet buffer will only attempt to read more frame data from the external memory when space is available in the internal packet buffer memory. If space is not available it must wait until a packet fetched by the MAC completes transmission and is subsequently removed from the packet buffer memory. Note that if full store and forward mode is active and if a single frame is fetched that is too large for the packet buffer memory, the frame is flushed and the DMA halted with an error status. This is because a complete frame must be written into the packet buffer before transmission can begin, and therefore the minimum packet buffer memory size should be chosen to satisfy the maximum frame to be transmitted in the application.

To ensure maximum throughput and full line-rate It is recommended that the transmit packet buffer is greater than twice the size of the largest frame that will be transmitted for the intended application. This is to ensure the MAC pipeline can be kept full at all times. If only short frames will ever be transmitted, it is recommended that an SRAM size substantially more than twice the size of the largest frame is used – this is to offset and smooth potential random latency delays on the AHB or AXI data fetches.

In full store and forward mode, once the complete transmit frame is written into the packet buffer memory, a trigger is sent across to the MAC transmitter, which will then begin reading the frame from the packet buffer memory. Since the whole frame is present and stable in the packet buffer memory an underflow of the transmitter is not possible. The frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be re-transmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from the AHB or AXI system memory.

In partial store and forward mode, a trigger is sent across to the MAC transmitter as soon as sufficient packet data is available in the internal buffer, which will then begin fetching the frame from the packet buffer memory. If, after this point, the MAC transmitter is able to fetch data from the packet buffer faster than the DMA can fill it, an underflow of the transmitter is possible. In this case, the transmission is terminated early, and the packet buffer is flushed. Transmission can only be restarted by writing to the transmit START bit, or by toggling the `trigger_dma_tx_start` input.

In half duplex mode, the frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be re-transmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from external system memory.

In full duplex mode, the frame is removed from the packet buffer on the fly

Other than underflow, the only MAC related errors that can occur are due to collisions during half duplex transmissions. When a collision occurs the frame still exists in the packet buffer memory so can be retried directly from there. Only once the MAC transmitter has failed to transmit after sixteen attempts is the frame finally flushed from the packet buffer.

Receive Packet Buffer:

The receive packet buffer stores frames from the MAC receiver along with their status and statistics. Frames with errors are flushed from the packet buffer memory, whilst good frames are pushed onto the DMA AHB interface.

The receiver packet buffer monitors the external FIFO write interface from the MAC receiver and translates the FIFO pushes into packet buffer writes. At the end of the received frame the status and statistics are buffered so that the information can be used when the frame is read out. When programmed in full store and forward mode, if the frame has an error the frame data is immediately flushed from the packet buffer memory allowing subsequent frames to utilise the freed up space. The status and statistics for bad frames are still used to update the GEM_GXL registers.

To accommodate the status and statistics associated with each frame, up to 4 words (one being for descriptor timestamp capture when enabled) (or 2 if you are configured in 64 bit datapath mode) per packet are reserved at the end of the packet data. If the packet was bad and requires to be dropped, the status and statistics are the only information held on that packet.

The receiver packet buffer will also detect a full condition such that an overflow condition can be detected. If this occurs subsequent packets will be dropped and an RX overflow interrupt is raised.

For full store and forward, the DMA will only begin packet fetches once the status and statistics for a frame are available. If the frame has a bad status due to a frame error, the status and statistics are passed onto the GEM_GXL registers. If the frame has a good status, the information is used to read the frame from the packet buffer memory and burst onto the AHB/AXI using the DMA buffer management protocol. Once the last frame data has been transferred to the FIFO, the status and statistics are updated to the GEM_GXL registers.

If partial store and forward mode is active, the DMA will begin fetching the packet data before the status is available. As soon as the status becomes available, the DMA will fetch this information before continuing to fetch the remainder of the frame. Once the last frame data has been transferred to the FIFO, the status and statistics are updated to the GEM_GXL registers.

Priority Queuing in the DMA

The DMA by default uses a single transmit and receive queue. This means the list of transmit/receive buffer descriptors point to data buffers associated with a single transmit/receive data stream. When the DMA is configured to use packet buffer memories, the GEM_GXL can optionally select up to 16 priority queues. Each queue has an independent list of buffer descriptors pointing to separate data streams. The maximum number of queues is determined by setting appropriate defines in GEM_GXL_defs.v (refer to the implementation application notes at the end of this document for details). By default, each queue (up to the maximum configured in GEM_GXL_defs.v) is active. Queues may be disabled by setting bit 0 of the Transmit or Receive Buffer Queue Base Address register, for which there is one per queue. Note that at least one queue must always remain enabled and only the top indexed queues may ever be disabled. For example for a system configured to have 12 queues, the user may decide that only 7 of these need to be active. In this case, the user would disable queues 8 to 12 by setting bit 0 of the Transmit or Receive Buffer Queue Base Address registers specific to queues 8-12.

In the transmit direction, higher priority queues are always serviced before lower priority queues. This strict priority scheme requires the user to ensure that high priority traffic is constrained such that lower priority traffic will have required bandwidth. The DMA will determine the next queue to service by initiating a sequence of buffer descriptor reads interrogating the ownership bits of each. The buffer descriptor corresponding to the highest priority queue is read first. If the ownership bit of this descriptor is set, then the DMA will progress to reading the 2nd highest priority queue's descriptor. If that ownership bit read of this lower priority queue is set, then the DMA will read the 3rd highest priority queue's descriptor, and so on. If all the descriptors return an ownership bit set, then a resource error has occurred, an interrupt is generated and transmission is automatically halted. Transmission can only be restarted by setting the START bit in the network control register or by toggling the `trigger_dma_tx_start` input. The DMA will need to identify the highest available queue to transmit from when the START bit in the network control register is written to and the TX is in a halted state, or when the last word of any packet has been fetched from external AHB or AXI memory.

The transmit DMA will maximize the effectiveness of priority queuing by ensuring that high priority traffic be transmitted as early as possible after being fetched from external memory.

High priority traffic will be pushed to the MAC layer depending on the specific transmit scheduling algorithm employed (refer to the section “transmit scheduling algorithm”). In order to facilitate the scheduler, the SRAM based packet buffer used within the transmit DMA is split into regions, one region per queue. The size of each region determines the amount of SRAM space allocated per queue and will be configurable by setting relevant defines in gem_gxl_defs.v file. This is implemented by first splitting the transmit SRAM into a number of segments of equal size. This number must be a power of two. Then, each queue is granted a number of segments. These configuration options and how to set them are explained further in the implementation application notes at the end of this document. Setting the configuration only sets the initial setting for SRAM queue allocation. It can be changed by the user by writing to the transmit segment allocation registers at offset 0x5a0 and 0x5a4. This allows the user to tweak the SRAM allocation for each queue and is very useful if the user wishes to disable queues and reallocate the memory assigned to an active queue. There are three bits per queue. Writing a value of 0 allocates 1 segment for the queue. A value of 1 would allocate 2 segments, a value of 2 would allocate 4 etc up to a maximum allocation of 32 segments (by writing a value of 5). Writing a vaue of 6 or 7 is not permitted.

For each queue, there is an associated Transmit Buffer Queue Base Address register. For the lowest priority queue (or the only queue when only 1 queue is selected), the Transmit Buffer Queue Base Address is located at address 0x1C. For all other queues, the Transmit Buffer Queue Base Address registers are located at sequential addresses starting at address 0x440.

In the receive direction each data packet is written to external AHB/AXI data buffers in the order that it is received. For each queue, there is an independent set of receive buffers for each queue. There is therefore a separate Receive Buffer Queue Base Address register for each queue. For the lowest priority queue (or the only queue when only one queue is selected), the Receive Buffer Queue Base Address is located at address 0x18. For all other queues, the Receive Buffer Queue Base Address registers are located at sequential addresses starting at address 0x480 for queues one to seven and from 0x5c0 for queues eight to fifteen. Every received packet will pass through a programmable screening algorithm which will allocate to that frame a particular queue to route it to. The user interface to the screeners is via two banks of programmable registers, screener type 1 match registers and screener type 2 registers.

Screener type 1 registers allow the user to route received frames based on particular IP and UDP fields extracted from the received frames. Specifically these fields are DS (Differentiated Services field of IPv4 frames), TC (Traffic class field of IPv6 frames) and/or the UDP destination port. These fields are compared against the values stored in the each of the screener type 1 match registers. If the result of this comparison is positive, then the received frame is routed to the priority queue specified in that screener type 1 register. The number of type 1 screeners is determined by a define in the gem defines file. Up to 16 screener type 1 registers are allocated address space in the default GEM address map, although this could be increased if necessary by redistributing other GEM registers. Refer to the type 1 screening register descriptions for further programming details.

Screener Type 2 match registers operate independently of screener type 1 registers and offer additional match capabilities, extending the capabilities into vendor specific protocols. The Type 2 screening allows a screen to be configured that is the combination of all or any of the following comparisons…

1) An Enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself.
2) An Enabled EtherType. The ethertype field inside the screener type 2 register maps to one of 8 ethertype match registers. The extracted ethertype is compared against the ethertype register designated by this ethertype field. The number of ethertype registers is configurable (up to 8) via the gem defines file.
3) An Enabled Field Compare A.
4) An Enabled Field Compare B.
5) An Enabled Field Compare C.

Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via a control bit in the associated compare word1. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the ethertype field, the byte following the end of the IP header (IPv4 or IPv6) or the byte following the end of the TCP/UDP header. Note the logic to decode the IP header or the TCP/UDP header is reused from the TCP/UDP/IP checksum offload logic and therefore has the same restrictions on use (the main limitation is that IP fragmentation is not supported). Refer to the Checksum Offload for IP, TCP and UDP section of this documentation for further details. The Compare Register field points to a single pool of 32 compare Registers. Compare A, Compare B, and Compare C use a common set of compare registers. The number of compare registers is configurable (up to 32) via the in gem_gxl_defs.v file.

Note compare A, B and C together allow matching against an arbitrary 48 bits of data and so can be used to match against a MAC address.

All enabled comparisons are ANDed together to form the overall type 2 screener match.

Refer to the type 2 screening register descriptions for further programming details.

The number of Type 2 screeners is determined by a define in the in gem_gxl_defs.v file. Up to 16 screener type 2 registers are allocated address space in the default GEM_GXL address map, although this could be increased if necessary by redistributing other GEM_GXL registers.

The following summarizes the available configurations that the GEM_GXL core can support with regards to priority queuing.

- There can be between one and 16 priority queues configured
- When the number of priority queues is greater than 1, the user has the option of including up to 16 type1 screeners, which allow the user to select the queue a receive packet will be routed to based on its UDP and/or TOS/TC fields.
- When the number of priority queues is greater than 1, the user has the option of including up to 16 type2 screeners, which allow the user to select the queue a receive packet will be routed to based on its VLAN, ethertype compare or user programmed compare fields.
- When the number of type2 screeners is greater than 1, the user has the option of including from zero up to 8 ethertype compare registers.
- When the number of type2 screeners is greater than 1, the user has the option of including from zero up to 32 user programmable field compare registers.
-

Each screener register is programmable via the APB interface. Although this is not recommended, it is possible that more than one screener register will be programmed to match against a single frame. If this happens there are 2 cases to consider …

1. If a received frame matches against multiple screeners of the same type, then the frame will route to the queue mapped by the screener located at the lowest numeric APB address. E.g. if screener type 2 #0 and screener type 2 #1 both match, then the frame will route to the queue identified in bits [3:0] of the screener type 2 #0 register.
2. If a received frame matches against a type 2 screener and a type 1 screener, then the type 1 screener will take precedence.

When a screener is matched, the received frame will be routed to a queue defined inside bits 3:0 of the screener register. Unmatched frames are routed to queue 0.

When the priority queuing feature is enabled, the number of interrupt outputs from the GEM_GXL core is increased to match the number of supported queues. The number of interrupt status registers are increased by the same number. Only DMA related events are reported using the individual interrupt outputs, as the GEM_GXL can relate these events to specific queues. All other events generated within the GEM_GXL are reported in the interrupt associated with the lowest priority queue. For the lowest priority queue (or the only queue when only 1 queue is selected), the interrupt status register is located at address 0x24. For all other queues, the interrupt status register is located at sequential addresses starting at address 0x400.

Advanced DMA Features to Reduce CPU Overhead

The DMA design has support for various optional offload features to reduce CPU overhead.

Receive Header/Data Splitting

Receive header data splitting is a feature which when enabled will force the receive DMA to split a received frame into its header and payload constituent parts. The header is not dropped, but instead separated from the payload and placed into its own DMA receive buffer. For example, a TCP/IPv4 frame will be split such that its Ethernet, IPv4 and TCP header is written into one or more buffers, and the TCP payload is written into one or more separate buffers. A status bit in the receive descriptors will identify whether the descriptor is pointing to a header buffer or a payload buffer. Another status bit will identify whether the descriptor is pointing to the last buffer of a header (the header may be split over multiple buffers). The length of the header will also be written in the descriptor that points to the last buffer of the header.

When header/data splitting is enabled, ALL received frames will have their L2/L3/L4 headers separated. Note that even standard Ethernet only frames containing just 14 bytes of header will be separated. Bit 5 of the DMA configuration register at offset 0x10 enables/disables header data splitting. When enabled there are specific bits of the DMA receive buffer descriptor (refer to the Receive Buffer Descriptor entry for details).

There are certain limitations to using this feature:

- Header Data Splitting applies to all received frames. Every frame received which includes a data payload will be split into at least 2 data buffers.
- Header Data Splitting is not currently available with the partial store and forward mode of operation. It should only be enabled when the full store and forward mode of operation is active.
-

Receive Side Coalescing

Receive segmentation coalescing is a mechanism to reduce CPU overhead. This is done by coalescing received TCP message segments together into a single large message. This means that when the message is complete the CPU only has to process the single header and act upon the one data payload. This will only be effective when processing small non signaling frames. For RSC to work the Header Data splitting feature must be enabled, and the priority queuing configuration options must be enabled (more details on this below).

The design will first inspect the low level received frames and identify if they are part of a particular TCP stream or not. This is performed using the existing priority queuing feature (refer to priority queuing section above). Each defined receive priority queue (other than queue 0) will map to a TCP stream capable of performing RSC. Therefore the priority queue screeners must be setup such that received frames can be inspected and routed to the appropriate queue. This can be achieved as follows:

Note it is assumed the reader is already familiar with the priority queue features as defined earlier in this document.

When the ethertype = IPv4 (0800) then the subsequent TCP/IP streams (or flows) are identified by the 4-tuple of IP source and destination address and TCP source and destination port. This is done using the type2 screeners. To provide sufficient compare functionality all three A, B, and C compares are required and the full 32-bit comparison (i.e. mask functionality disabled) will be required for each comparison. The diagram below gives an example of the offsets required to identify the tuples.

The coalesced frames should be controlled using multiple queues therefore RSC requires the following to be setup:

- Hardware Configuration
- Memory Setup
- Enable RSC Control

Hardware configuration requirements

RSC functionality is only supported when using an AXI interface therefore the following should be set

```
`define gem_axi
```

RSC functionality is only available if the module is configured to have at least 2 queues. Multiple queues is a hardware configuration and is set using a `define prior to compile.

```
`define dma_priority_queue1
```

The following must be enabled for RSC.

```
`define gem_pbuf_rsc
```

RSC needs a number of type2 screeners defined. The number chosen should at minimum reflect the number of TCP streams required. 3 Compare registers per screener are also required.

```
`define num_type2_screeners 8'd3
```

Memory Setup

Descriptor buffers point to the main system memory, this area in the memory should have buffers set to the maximum size of 16K. As a result we would also need to set the mac's max frame size to less than 16K. As header data splitting is enabled software must provide at least two descriptors the first being for the header followed by a second for the payload. This will result in a waste of memory space since the header will never be 16K therefore it is suggested that software overlaps the second descriptor as shown in the diagram below.

The design will coalesce buffers up to the maximum size of 16K. If the next packet causes the coalesced buffer to exceed the maximum size, at this point it will close the buffer and restart a new buffer. Closing the buffer involves updating the IP length, checksum and ACK fields in the header buffer and writing used bits for the header and payload buffers. The ACK field is written with the most recently received ACK number with the ACK flag set.

If the next packet has TCP Flags SYN, FIN, RST or URG set, or a frame is received with an out of order TCP sequence number, these events are treated as “RSC STOP” events. In these cases, the user can optionally select via the RSC control register offset 0x58 whether coalescing should cease (restarting only upon software setting up a new TCP stream for coalescing) or whether the design should simply restart coalescing into a new buffer on the frame following the "RSC STOP" event. The frame received that caused the "RSC STOP" event is always forwarded to main memory in isolation (non coalesced).

If the next packet has the PSH TCP Flag set, this will also be treated as an "RSC STOP" event. However, coalescing will restart on the next frame received associated with the TCP stream.

The memory size for each priority queue is defined in the dma_rxbuf_size_q1 registers. For coalescing on a particular queue, these should be set to the maximum size of 16K.

Control

Header Data splitting must be enabled prior to enabling RSC. RSC is enabled via the RSC Control register offset 0x58. Via this register, the number of TCP streams for coalescing is chosen (each priority queue will map to a particular TCP stream). There is also a control bit in this register to allow the user to select whether coalescing should cease after the "TCP STOP" event occurs.

RSC Limitations

- Little Endian only
- Header Data Splitting must be enabled to use RSC.
- Buffer Sizes must be set to 16K.
- RSC only works with IPv4 headers.
- RSC should not be used if the controller will receive jumbo frames.
- Currently RSC cannot be used in conjunction with partial store and forward mode of operation.
- When the gem_pbuf_rsc define has been set the receive buffer offset cannot be changed in the network configuration register

Large Send Offload

Large Send Offload functionality is included to reduce CPU overhead for transmit. Two mechanisms are included -TCP Segmentation Offload (TSO) and UDP Fragmentation Offload (UFO). CPU overhead is reduced by permitting software to handoff TCP and UDP frames which are larger than the current TCP Maximum Segment Size (MSS) and Ethernet Maximum Frame Size (MFS). Hardware splits the large software supplied frames into smaller frames for transmission and modifies the header fields as required. TCP offload is achieved by creating smaller segments at the TCP level. UDP offload is achieved by performing fragmentation at the IP level.

For correct TSO and UFO functionality hardware checksum insertion must be enabled (DMA Configuration Register, bit 11).

For both TSO and UFO, software must supply the header and payload of the large frame in at least two buffers (one for header and one or more for payload).Interaction between hardware and software is similar to the use case where multiple buffers are used to supply a single Ethernet frame.

There are no control registers associated with TSO and UFO – the features are enabled and configured via fields in the descriptors for the header and payload buffers.

TSO Operation

Software supplies a large frame containing a large TCP segment via a header buffer and one or more payload buffers, and supplies the associated header and payload descriptor(s). The header buffer contains the Ethernet, IPv4 or IPv6 and TCP headers. The header descriptor has bits [18:17] set to 2'b10 or 2'b11 to enable TSO.

The header descriptor contains a sequence number select bit at bit 19. This is cleared if the hardware will use the sequence number from the header buffer for the first small segment (and generate appropriate sequence numbers for the other small segments). It is set if hardware will generate sequence numbers for all of the small segments. In this case the sequence number for the first small segment will be related to the most recent previously transmitted segment of the same stream that were transmitted with TSO enabled. The hardware supports 4 separate TCP streams per priority queue (and implements a sequence number counter for each). The TCP stream is identified by bits [25:24] of the header descriptor.

The payload descriptor contains a TCP Maximum Segment Size (MSS) field at bits [29:16] and hardware uses this MSS value to process the payload buffer. Each payload buffer is processed separately (i.e. an integer number of small segments is generated from each payload buffer) using the MSS value in the associated descriptor. Software may supply different MSS values in the payload descriptors of a large frame which has 2 or more payload buffers.

Hardware will load the header descriptor and store it internally, and will use it to read the associated header buffer multiple times as the payload buffers are processed and small segments are transmitted. Hardware will modify the IPv4 Total Length, IPv4 Header Checksum, IPv6 Payload Length, TCP Sequence Number and TCP Checksum fields. Software may modify the TCP Acknowledgement Number and TCP Window Size values in the header buffer during the hardware segmentation process – however as software has no notification of when hardware is reading the header buffer care must be taken to avoid hardware reading a partially updated value. This can be achieved by locating the base address of the header buffer so that a software writes of a field to be modified is achieved via a single AXI write access beat – i.e. the Acknowledgement Number field is on a 32-bit boundary in memory and the Window Size field is on a 16-bit boundary in memory. Placement of the header buffer base address to achieve this is dependent on the Ethernet header length.

Hardware discards the IPv4 Total Length and IPv6 Payload length values in the header buffer supplied by software and therefore it is possible for software to supply a very large TCP segment that exceeds the limitations imposed by IP encapsulation.

Hardware will perform a write back to the header descriptor when the data contained in a payload buffer has been transmitted and the associated payload descriptor has the last bit set.

Software can make use of the sequence number generation functionality for all TCP segments regardless of size by delivering the frame in separate header and payload buffers and enabling TSO in the header descriptor – it is not necessary for the supplied segment to exceed the TCP Maximum Segment Size in order to use hardware sequence number functionality.

TSO Limitations

- Ethernet Header extensions are limited to VLAN and Stacked VLAN
- Payload descriptors must not have a value of zero in the length field
- The Ethernet frame supplied by software must not contain IP fragments

UFO Operation

Software supplies a large frame containing a large UDP datagram via a header buffer and one or more payload buffers, and supplies the associated header and payload descriptor(s). The header buffer contains the Ethernet and IPv4 headers. The header descriptor has bits [18:17] set to 2'b01 to enable UFO. The payload buffer or buffers contain the IP payload, which consists of the UDP header and UDP payload.

The payload descriptor contains an Ethernet Maximum Frame Size (MFS) field at bits [29:16] and hardware will use this MFS value to process the payload buffer. Each payload buffer is processed separately (i.e. an integer number of fragments is generated from each payload buffer) using the MFS value in the associated descriptor. Software may supply different MFS values in the payload descriptors of a large frame which has 2 or more payload buffers. When software supplies two or more payload buffers all payload buffers apart from the last must be a multiple of 8 bytes in size.

Hardware will load the header descriptor and store it internally, and will use it to read the associated header buffer multiple times as the payload buffers are processed and fragments are transmitted. Hardware will modify the IPv4 Total Length, IPv4 More Fragments Flag and IPv4 Fragment Offset fields. Hardware will not calculate the UDP checksum or modify the UDP checksum field. Therefore software must set a value of zero in the checksum field in the UDP header (in the first payload buffer) to indicate to the receiver that the UDP datagram does not include a checksum.

Hardware will perform a write back to the header descriptor when the data contained in a payload buffer has been transmitted and the associated payload descriptor has the last bit set.

UFO Limitations

- Ethernet Header extensions are limited to VLAN and Stacked VLAN
- IP encapsulation is limited to IPv4
- Payload descriptors must not have a value of zero in the length field
- Payload buffers, apart from those with the last bit set in the associated descriptor, must be a multiple of 8 bytes in length
- The Ethernet frame supplied by software must not contain IP fragments

External FIFO Interface

Description

In applications where the DMA operation is not required, an exposed FIFO interface is available for both the transmit and receive data paths. Data bus widths in each direction can be configured to be either 32-bit, 64-bit or 128-bit. The `dma_data_width` bits (bits 21 and 22 of the network configuration register) determine the bus width used within the MAC.

The `tx_r_data_rdy` signal indicates to the MAC that there is sufficient data in the FIFO for transmission to commence. Once this signal becomes active, the transmit module initiates a read cycle by asserting `tx_r_rd` for one `tx_clk` cycle. The FIFO should indicate valid data at the FIFO interface by asserting `tx_r_valid` for a single cycle. The latency between the read and valid data is controlled using the `tx_r_valid` response, which may be returned during the same cycle as the `tx_r_rd` request. Once a read has commenced it must be terminated with `tx_r_valid` or `tx_r_underflow` even if `tx_r_data_rdy` is de-asserted.

The MAC transmitter searches for start of packet (SOP), indicated by `tx_r_sop`, and transmission commences once this input becomes valid coincident with `tx_r_valid`. The MAC will continuously search for `tx_r_sop` whilst `tx_r_data_rdy` is set. Once the SOP has been read, data is extracted from the FIFO on the `tx_r_data` input every time `tx_r_valid` is set. The MAC continues to read the frame from memory using `tx_r_rd` and transmission takes place.

The end of frame is indicated by `tx_r_eop` being set coincident with `tx_r_valid`. Once this condition occurs, the `tx_r_mod` input indicates how many data bytes are valid in the last transfer. `tx_r_mod` is only valid at the end of the frame, and cannot be used to insert an offset at the start of frame.

For frames smaller than the data width, both the SOP and end of packet (EOP) indicators must be set in the same data transfer.

If two SOPs occur with no intervening EOP then there is an underrun and both frames are lost, unless the second SOP occurs on the same cycle as EOP in which case the second SOP is ignored. A properly configured system should not generate two SOPs with no intervening EOP.

The `tx_r_err` signal may be asserted at any stage in the frame, being driven coincident with `tx_r_valid` being set.

In applications where the external FIFO interface is required to operate in a half duplex system, `tx_r_status` information is available indicating where collisions, excess collisions, late collisions and under runs have occurred. Upon each of these conditions, it is necessary to flush the external FIFO and the MAC must wait for this operation to complete before commencing further frames. A falling edge on the `tx_r_flushed` input signal indicates when the flush is complete. If a collision occurs, it is necessary for the external FIFO interface to repeat the transfer of the current frame, so that the frame may be successfully re-transmitted.

`tx_r_status` information must be acknowledged by the external FIFO interface by toggling the `tx_r_status_tog` input to the MAC each time status is taken. This causes the `tx_r_status` bus to be cleared until the next end of frame or collision occurs.

For reception, once it has been determined that a frame should be written to memory, the MAC receiver writes data to the FIFO using `rx_w_wr` and `rx_w_data`. SOP is indicated by `rx_w_sop` and EOP using `rx_w_eop`. For the last transfer of a frame, the `rx_w_mod` is driven to indicate how many bytes are valid in this last transfer. rx_w_sop and rx_w_eop will not be asserted in the same cycle.

The `rx_w_err` signal is set when the MAC encounters a reception error such as frame too short or CRC error. An `rx_w_status` bus is available giving status about the frame being received, such as frame length, matched internally or externally, broadcast, multicast, etc. The `rx_w_eop` signal is always asserted in the same cycle as `rx_w_err`.

The `rx_w_overflow` input signal can be asserted in the `rx_clk` domain when an external FIFO fails to receive a frame from the GEM FIFO interface. If `rx_w_overflow` is asserted sometime between the SOP and EOP writes, the remainder of the packet will continue to be written out, but at the end of the packet `rx_w_err` will be asserted together with `rx_w_eop`. Additionally, if `rx_w_overflow` is asserted then the receive statistics registers will not count the frame as good. `rx_w_overflow` must be asserted one cycle after `rx_w_eop` or earlier.

`rx_w_flush` is asserted when the GEM receive path is disabled in the network control register. If you disable frame reception while a frame is being transferred on the FIFO interface you will not see an `rx_w_eop` indication. `rx_w_flush` is an rx_clk timed signal that you can use to clear the external receive FIFO when receive is disabled.

Priority Queuing without the DMA

Priority Queuing is supported via the FIFO interface. By default there is a single transmit and receive queue. The GEM_GXL can optionally select up to 16 priority queues, determined by setting appropriate defines in GEM_GXL_defs.v (refer to the implementation application notes at the end of this document for details). When there is more than one queue, the external FIFO interface is extended support queue ID signals. The width of `tx_r_data_rdy` and `tx_r_rd` is extended such that the width matches the number of configured queues. However, the overall functionality of these signals does not change. While multiple bits of `tx_r_data_rdy` may be set simultaneously to indicate there is sufficient data in the various external FIFOs (one per queue) for transmission to commence, the GEM_GXL will only ever set one bit of `tx_r_rd` at any one time. It is for the integrator to multiplex in the various external FIFO's to interoperate with this signalling. For the receiver, a new output will be presented to the receive FIFO interface to indicate the queue ID the receive frame is destined. This will be set as soon as the GEM_GXL can identify the queue and therefore is dependent on how the packet identification is programmes, which is described below:

Every received packet will pass through a programmable screening algorithm which will allocate to that frame a particular queue to route it to. The user interface to the screeners is via two banks of programmable registers, screener type 1 match registers and screener type 2 registers.

Screener type 1 registers allow the user to route received frames based on particular IP and UDP fields extracted from the received frames. Specifically these fields are DS (Differentiated Services field of IPv4 frames), TC (Traffic class field of IPv6 frames) and/or the UDP destination port. These fields are compared against the values stored in the each of the screener type 1 match registers. If the result of this comparison is positive, then the received frame is routed to the priority queue specified in that screener type 1 register. The number of type 1 screeners is determined by a define in the gem defines file. Up to 16 screener type 1 registers are allocated address space in the default GEM_GXL address map, although this could be increased if necessary by redistributing other GEM_GXL registers. Refer to the type 1 screening register descriptions for further programming details.

Screener Type 2 match registers operate independently of screener type 1 registers and offer additional match capabilities, extending the capabilities into vendor specific protocols. The Type 2 screening allows a screen to be configured that is the combination of all or any of the following comparisons…

6) An Enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself.
7) An Enabled EtherType. The ethertype field inside the screener type 2 register maps to one of 8 ethertype match registers. The extracted ethertype is compared against the ethertype register designated by this ethertype field. The number of ethertype registers is configurable (up to 8) via the gem defines file.
8) An Enabled Field Compare A.
9) An Enabled Field Compare B.
10) An Enabled Field Compare C.

Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via a control bit in the associated compare word1. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the ethertype field, the byte following the end of the IP header (IPv4 or IPv6) or the byte following the end of the TCP/UDP header. Note the logic to decode the IP header or the TCP/UDP header is reused from the TCP/UDP/IP checksum offload logic and therefore has the same restrictions on use (the main limitation is that IP fragmentation is not supported). Refer to the Checksum Offload for IP, TCP and UDP section of this documentation for further details. The Compare Register field points to a single pool of 32 compare Registers. Compare A, Compare B, and Compare C use a common set of compare registers. The number of compare registers is configurable (up to 32) via the in gem_gxl_defs.v file.

Note compare A, B and C together allow matching against an arbitrary 48 bits of data and so can be used to match against a MAC address.

All enabled comparisons are ANDed together to form the overall type 2 screener match.

Refer to the type 2 screening register descriptions for further programming details.

The number of Type 2 screeners is determined by a define in the in gem_gxl_defs.v file. Up to 16 screener type 2 registers are allocated address space in the default GEM_GXL address map, although this could be increased if necessary by redistributing other GEM_GXL registers.

The following summarizes the available configurations that the GEM_GXL core can support with regards to priority queuing.

- There can be between one and 16 priority queues configured
- When the number of priority queues is greater than 1, the user has the option of including up to 16 type1 screeners, which allow the user to select the queue a receive packet will be routed to based on its UDP and/or TOS/TC fields.
- When the number of priority queues is greater than 1, the user has the option of including up to 16 type2 screeners, which allow the user to select the queue a receive packet will be routed to based on its VLAN, ethertype compare or user programmed compare fields.
- When the number of type2 screeners is greater than 1, the user has the option of including from zero up to 8 ethertype compare registers.
- When the number of type2 screeners is greater than 1, the user has the option of including from zero up to 32 user programmable field compare registers.

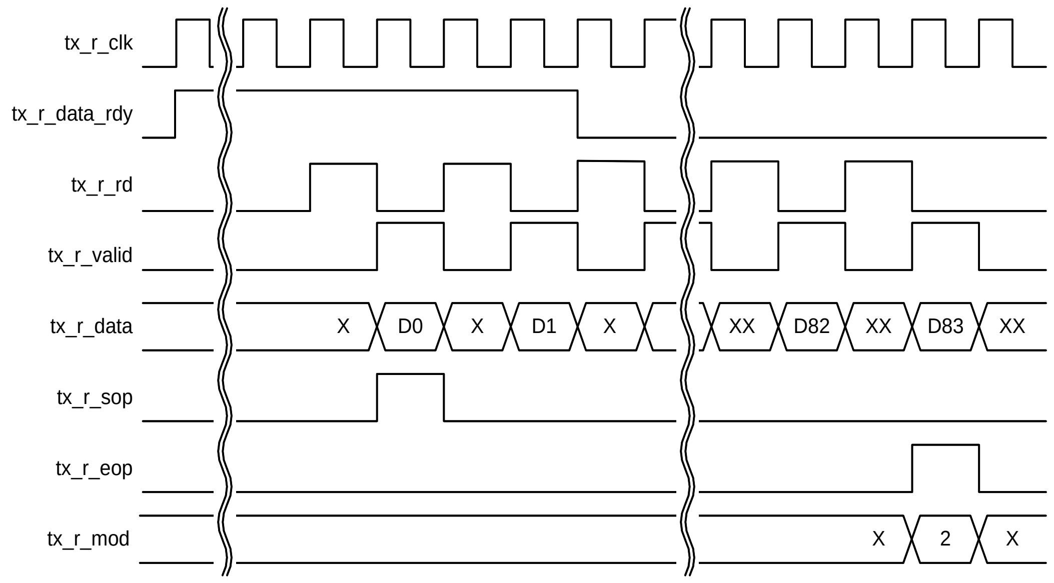
Each screener register is programmable via the APB interface. Although this is not recommended, it is possible that more than one screener register will be programmed to match against a single frame. If this happens there are 2 cases to consider …

3. If a received frame matches against multiple screeners of the same type, then the frame will route to the queue mapped by the screener located at the lowest numeric APB address. E.g. if screener type 2 #0 and screener type 2 #1 both match, then the frame will route to the queue identified in bits [3:0] of the screener type 2 #0 register.
4. If a received frame matches against a type 2 screener and a type 1 screener, then the type 1 screener will take precedence.

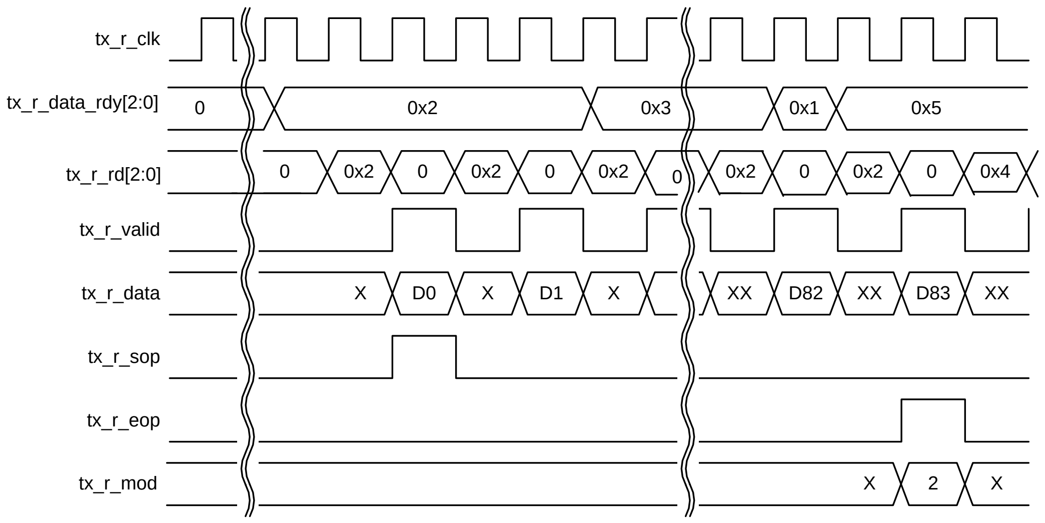
When a screener is matched, the received frame will be routed to a queue defined inside bits 3:0 of the screener register. Unmatched frames are routed to queue 0.

External FIFO Interface Timing Criteria

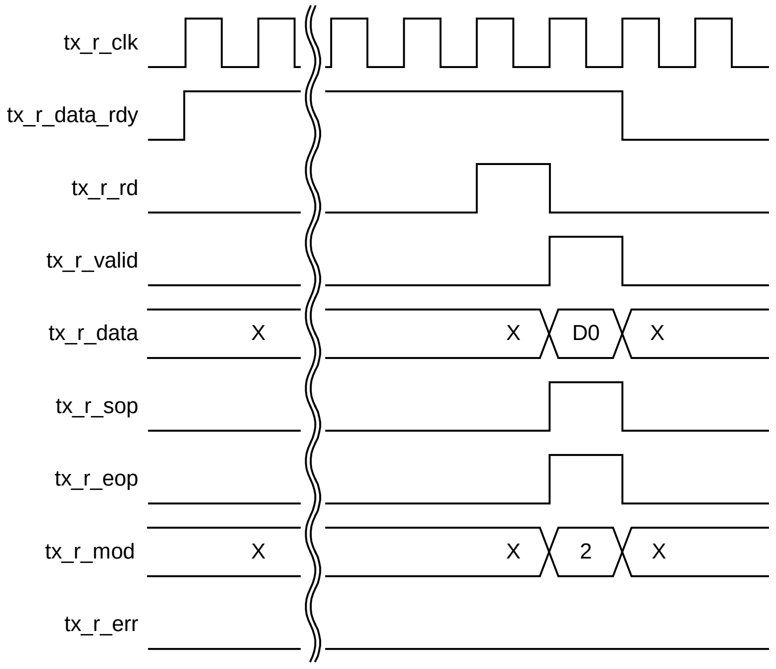
The following diagram shows detailed timing relationships for a frame on the external FIFO interface (MAC transmit) with one cycle between read request and data valid. Only one priority queue is shown here.



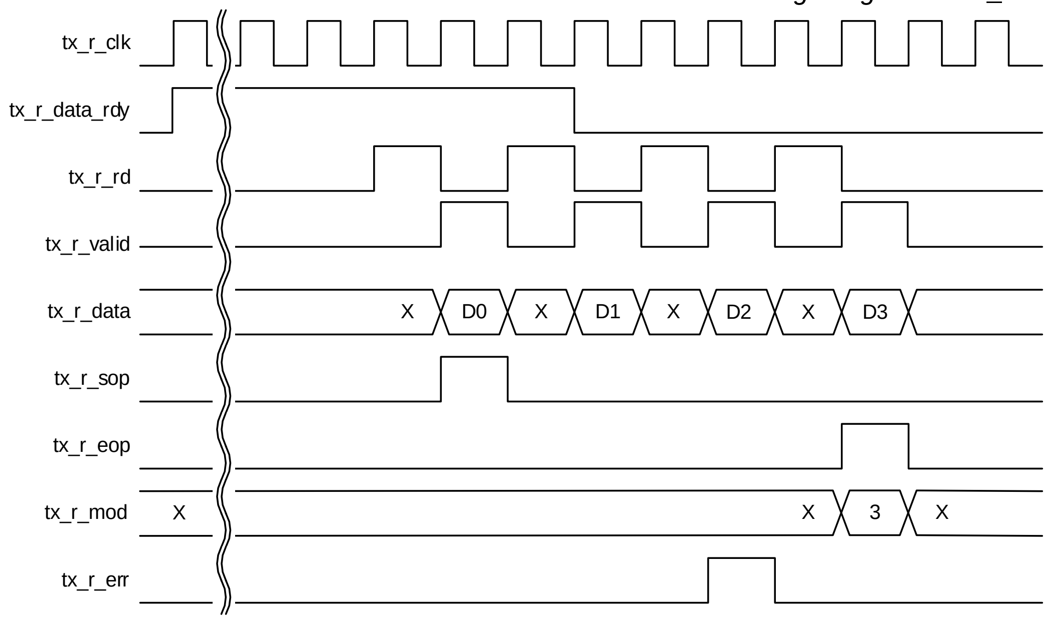
A similar diagram showing a 3 queue system is shown below :



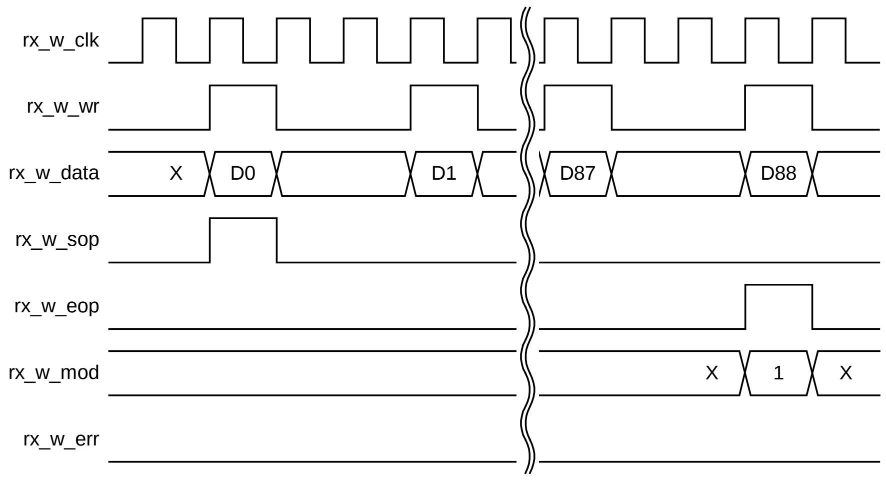
The following diagram shows detailed timing relationships for a frame on the external FIFO interface (MAC transmit) that incorporates a 2-byte frame with a SOP and EOP in the same transfer.



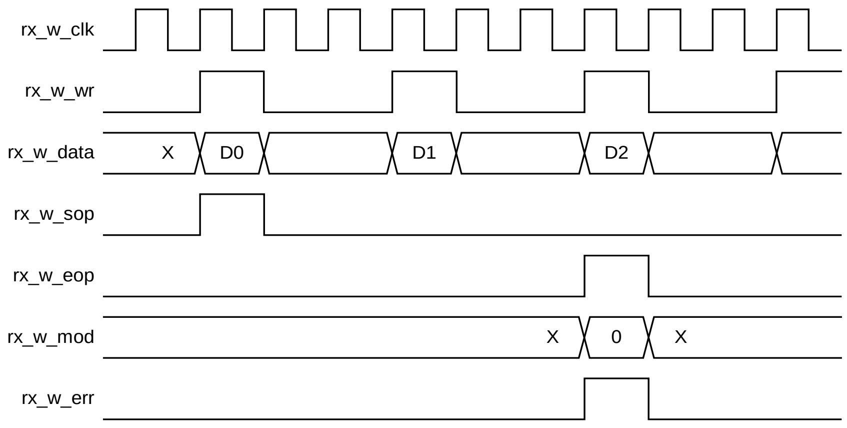
The following diagram shows detailed timing relationships for a frame on the external FIFO interface (MAC transmit) with frame error.



The following diagram shows a frame on the external FIFO interface (MAC receive).



The following diagram shows a frame on the external FIFO interface (MAC receive) with frame error.



Additional Low Latency TX FIFO Interface for DMA configurations

When the design is configured to include the SRAM based packet buffer DMA, it is possible for the user to additionally add an extra low latency FIFO TX host interface to inject high priority fixed latency traffic directly into the transmit path of the MAC bypassing the DMA altogether. This additional TX FIFO host interface is optionally included by defining `gem_tx_add_fifo_if verilog define. Packets that are injected into the MAC via the low latency FIFO interface have the same latency as the external fifo_interface mode, plus one tx_clk clock cycle, when no DMA frame is being sent or from the end of the current DMA frame ends, and is useful if particular packets needing a fixed transmit latency are required. DMA and low latency TX frames are dynamically arbitrated with low latency TX FIFO interface frames being highest priority. If a frame from the DMA is being transmitted to the MAC, it will complete before the Low Latency TX frame is sent but following DMA frames will be delayed if required.

As there is no RX FIFO interface in this mode all received frames are passed to the RX DMA.

When using the additional low latency tx fifo the following restrictions apply:

- Half Duplex is not supported.
- No register status updating for packet complete for fifo frames

Underflow errors that occur when packets are being fetched via the external FIFO interface are not reported in the status registers or interrupts. Underflow status will still get reported via the FIFO interface outputs.

Host Interface Soft Select Configuration External FIFO / DMA interfaces

The design can be configured to have both external FIFO interface and the SRAM packet buffer DMA interfaces. This can optionally be built by defining `define gem_host_if_soft_select.

In this mode either the external fifo interface or the DMA interface can be selected. They cannot be used together, the interface selection is a static programming option that should be setup before enabling the RX and TX paths.

Transmit Scheduling Algorithm

When multiple priority queues have been selected, the transmit scheduler is automatically included in the design and is responsible for selecting the next queue to be serviced either from the attached DMA or from the external FIFO interface when the DMA is not included. There are four scheduling algorithms available to the user and, with some exceptions detailed further below, the user can select one of the four modes for each queue.

- 802.1Qav Support – Credit Based Shaping
  A credit-based shaping algorithm is available on the two highest priority active queues and is defined in 802.1Qav: Forwarding and Queuing Enhancements for Time-Sensitive Streams. Traffic shaping is enabled via the CBS (Credit Based Shaping) Control register (0x4bc) or the general transmit scheduling control register (offset 0x580). These two registers are aliased, so updating one register will automatically update the other. Note it is permitted to enable CBS only on the second highest priority queue and not on the highest, in which case the highest priority queue would always take precedence.

  Enabling CBS on a queue will enable a counter which stores the amount of transmit 'credit', measured in bytes that a particular queue has. A queue may only transmit if it has non-negative credit. If a queue has data to send, but is held off from doing as another queue is transmitting, then credit will accumulate in the credit counter at the rate defined in the IdleSlope register for that queue. IdleSlope is the rate of change of credit when waiting to transmit and must be less than the value of the portTransmitRate. When this queue is transmitting the credit counter is decremented at the rate of sendSlope which is defined as the portTransmitRate – IdleSlope. A queue can accumulate negative credit when transmitting which will hold off any other transfers from that queue until credit returns to a non-negative value. No transfers are halted when a queue's credit becomes negative, it will accumulate negative credit until the transfer completes.

To ensure that the CBS scheduling is completely accurate a single transmit buffer should be used per Ethernet frame (rather than multi-buffer transmit frames).

- Fixed Priority.
  Any of the active queues can be selected as fixed priority and this is the default mode of operation for all queues. The queue index is used as the priority, where a higher index will have a higher priority than a lower index. The scheduler will always attempt to transmit from fixed priority queues with the highest priority. I.e. A fixed priority queue with a high queue index will always take precedence over a priority queue with a lower index.

- Deficit Weighted Round Robin (DWRR)
  Any of the active queues can be selected as DWRR. If DWRR is required, then at least two of the active queues should be selected as DWRR. It should not be used in conjunction with ETS, as both algorithms operating together would make little practical sense. A DWRR enabled queue has lower priority than a CBS enabled queue or a fixed priority queue with a higher index.

  The DWRR algorithm works by scanning all non-empty queues in sequence. Each queue is allocated a 'deficit counter' and an 8-bit weighting (or quantum) value. The value of the deficit counter is the maximum number of bytes that can be sent at the current time. If the deficit counter of the scanned queue is greater than the length of the packet wating for transmission then the packet will be transmitted and the value of the defict counter is decremented by the packet size. If it is not greater the scheduler will skip to the next DWRR enabled queue. If there is insufficient credit to transmit, the queue is simply skipped. If the queue is empty, the value of the deficit counter is reset to 0. If all queues have insufficient credit then each tx_clk cycle every queue's deficit counter is incremented by its quantum value until a queue's deficit counter obtains sufficient credit to transmit its first queued frame. The higher the quantum value chosen the quicker deficit counter will reach the required value. If all DWRR queues have the same weighting, then all queues will be granted the same overall bandwidth. The weighting value is stored in four programmable registers starting at offset 0x590. Since the design supports up to 16 priority queues, four physical registers are required. Refer to the register descriptions for further details.

  Note that if fixed priority queues are to be used in conjunction with DWRR, the fixed priority queues must be at a higher index value than the DWRR queues. A consequence of this is that the enabled DWRR queues shall form a contiguous set of queues starting from queue 0.

  If CBS is also used in conjunction with DWRR, the DWRR queues will share the remaining bandwidth after the CBS allocation has been deducted.

  Note that if transmit cut-thru is should not be enabled if the transmit scheduler is used.

- Enhanced Transmission Selection (ETS)
  The ETS algorithm is defined in 802.1Qaz: Enhanced Transmission Selection for Bandwidth Sharing between Traffic and allows traffic on specific queues to be bandwidth limited. Any of the active queues can be selected as ETS. If ETS is required, then at least two of the active queues should be selected as ETS. It should not be used in conjunction with DWRR as both algorithms operating together would make little practical sense. An ETS enabled queue has lower priority than a CBS enabled queue or a fixed priority queue with a higher index.

  For each ETS enabled queue, the user should program the bandwidth requirement for each queue as a percentage of total bandwidth (an 8-bit register is used and the sum of values programmed should not exceed decimal 100). This will be the maximum bandwidth to be granted to that queue. The actual scheduling algorithm operates in a round-robin style from lowest indexed queues up to the highest indexed queue in sequence. The bandwidth allocation percentage is stored in four programmable registers starting at offset 0x590 – these are the same registers used for DWRR. Since the design supports up to 16 priority queues, four physical registers are required. Refer to the register descriptions for further details.

If CBS is also used in conjunction with ETS, the sum of the ETS queue percentages should equal the remaining bandwidth after the CBS allocation has been deducted.

Note that if transmit cut-thru is should not be enabled if the transmit scheduler is used.

MAC Transmit Block

The MAC transmitter can operate in either half duplex or full duplex and transmits frames in accordance with the Ethernet IEEE 802.3 standard. In half duplex mode, the CSMA/CD protocol of the IEEE 802.3 specification is followed.

A small input buffer receives data through the external FIFO interface (from either the DMA module or external to the IP Core) which, depending on the `dma_bus_width` control bits in the network configuration register, will extract data in either 32-bit, 64-bit or 128-bit form. All subsequent processing prior to the final output is performed in bytes.

Transmit data can be output using the GMII/MII interface or through the TBI.

If the TBI is selected (gigabit and SGMII modes only), then the MAC transmitter passes 8-bit data to the PCS sub-layer for further processing prior to output on the TBI. In 10 and 100Mbps SGMII mode the MAC is clocked slower than the PCS which has the effect of the PCS sampling the same 8-bit data 10 or 100 times.

Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from the transmit FIFO interface a word at a time. When the GEM is configured for gigabit operation, the data output to the PHY uses all 8 bits of the `txd[7:0]` output. In 10/100 mode, transmit data to the PHY is nibble wide and least significant nibble first using `txd[3:0]` with `txd[7:4]` tied to logic 0.

If necessary, padding is added to take the frame length to 60 bytes. CRC is calculated using an order 32 bit polynomial. This is inverted and appended to the end of the frame taking the frame length to a minimum of 64 bytes. If the no CRC bit is set in the second word of the last buffer descriptor of a transmit frame neither pad nor CRC are appended. The no CRC bit can also be set through the FIFO interface.

In full duplex mode (at all data rates), frames are transmitted immediately. Back to back frames are transmitted at least 96 bit times apart to guarantee the interframe gap.

In half duplex mode, the transmitter checks carrier sense. If asserted, the transmitter waits for the signal to become inactive, and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter will transmit a jam sequence of 32 bits taken from the data register and then retry transmission after the back off time has elapsed. If the collision occurs during either the preamble or SFD, then these fields will be completed prior to generation of the jam sequence.

The back off time is based on an XOR of the 10 least significant bits of the data coming from the transmit FIFO interface and a 10 bit pseudo random number generator. The number of bits used depends on the number of collisions seen. After the first collision 1 bit is used, then the second 2 bits and so on up to the maximum of 10 bits. All 10 bits are used above ten collisions. An error will be indicated and no further attempts will be made if 16 consecutive attempts cause collision. This operation is compliant with the description in Clause 4.2.3.2.5 of the IEEE 802.3 standard which refers to the truncated binary exponential back off algorithm.

In 10/100 mode, both collisions and late collisions are treated identically, and back off and retry will be performed up to 16 times. When operating in gigabit mode, late collisions are treated as an exception and transmission is aborted, without retry. This condition is reported in the transmit buffer descriptor word 1 (late collision, bit 26) and also in the transmit status register (late collision, bit 7). An interrupt can also be generated (if enabled) when this exception occurs, and bit 5 in the interrupt status register will be set.

In all modes of operation, if the transmit DMA under runs, a bad CRC is automatically appended using the same mechanism as jam insertion and the `tx_er` signal is asserted. For a properly configured system this should never happen and also it is impossible if configured to use the DMA with packet buffers, as the complete frame is buffered in local packet buffer memory.

By setting when bit 28 is set in the network configuration register the IPG may be stretched beyond 96 bits depending on the length of the previously transmitted frame and the value written to the IPG_STRETCH register. The least significant 8 bits of the IPG_STRETCH register multiply the previous frame length (including preamble) the next significant 8 bits (+1 so as not to get a divide by zero) divide the frame length to generate the IPG. IPG stretch only works in full duplex mode and when bit 28 is set in the network configuration register. The IPG_STRETCH register cannot be used to shrink the IPG below 96 bits.

If the back pressure bit is set in the network control register or if the half_duplex_flow_control_en input is set in 10M or 100M half duplex mode, the transmit block transmits 64 bits of data, which can consist of 16 nibbles of 1011 or in bit rate mode 64 1s, whenever it sees an incoming frame to force a collision. This provides a way of implementing flow control in half duplex mode. Note this feature is not available in gigabit half duplex mode.

MAC Receive Block

These control bits are expected to be static and are not expected to be updated without disabling the RX path by clearing `bit[2]`, enable receive, of the network control register first.

All processing within the MAC receive block is implemented using 16 bit data path. The MAC receive block checks for valid preamble, FCS, alignment and length, presents received frames to the external FIFO interface (to either the DMA module or external to the IP core) and stores the frames destination address for use by the address checking block.

When the TBI is selected, 16 bit words are passed directly from the PCS layer to the MAC receiver, except in 10 and 100Mpbs SGMII modes when 8 bits are read. In 10 and 100Mpbs SGMII modes the MAC `rx_clk` runs more slowly than the PCS receive datapath clock and the same 8 bits of data is read 5 or 50 times.

If, during frame reception, the frame is found to be too long, a bad frame indication is sent to the external FIFO interface. The receiver logic ceases to send data to memory as soon as this condition occurs.

At end of frame reception the receive block indicates to the DMA block whether the frame is good or bad. The DMA block will recover the current receive buffer if the frame was bad.

Ethernet frames are normally stored in DMA memory or to the external FIFO interface complete with the FCS. Setting the FCS remove bit in the network configuration (bit 17) causes frames to be stored without their corresponding FCS. The reported frame length field is reduced by four bytes to reflect this operation.

The receive block signals to the register block to increment the alignment, CRC (FCS), short frame, long frame, jabber or receive symbol errors when any of these exception conditions occur.

If bit 26 is set in the network configuration CRC errors will be ignored and CRC errored frames will not be discarded, though the Frame Check Sequence Errors statistic register will still be incremented. Additionally if configured to use the DMA and not enabled for jumbo frames mode, then bit[13] of the receiver descriptor word 1 will be updated to indicate the FCS validity for the particular frame. This is useful for applications such as EtherCAT whereby individual frames with FCS errors must be identified.

Received frames can be checked for length field error by setting the length field error frame discard bit of the network configuration register (bit-16). When this bit is set, the receiver compares a frame's measured length with the length field (bytes 13 and 14) extracted from the frame. The frame is discarded if the measured length is shorter. This checking procedure is for received frames between 64 bytes and 1518 bytes in length.

Each discarded frame is counted in the 10 bit length field error statistics register. Frames where the length field is greater than or equal to 0x0600 hex will not be checked.

When operating in gigabit mode (half duplex), the receiver will discard frames which do not meet the minimal slot time of 512 bytes. If a burst is detected, the first frame is checked to ensure it meets the slot time, but all subsequent frames of the burst are checked to ensure they meet the minimum frame size of 64 bytes.

Checksum Offload for IP, TCP and UDP

The GEM_GXL can be programmed to perform IP, TCP and UDP checksum offloading in both the receive and transmit direction which is enabled by setting bit 24 in the network configuration register for receive and bit 11 in the DMA configuration register for transmit.

IPv4 packets contain a 16-bit checksum field, which is the 16-bit 1's complement of the 1's complement sum of all 16-bit words in the header. TCP and UDP packets contain a 16-bit checksum field, which is the 16-bit 1's complement of the 1's complement sum of all 16-bit words in the header, the data and a conceptual IP pseudo header.

To calculate these checksums in software requires each byte of the packet to be processed. For TCP and UDP this can use a large amount of processing power. Offloading the checksum calculation to hardware can result in significant performance improvements.

For IP, TCP or UDP checksum offload to be useful, the operating system containing the protocol stack must be aware that this offload is available so that it can make use of the fact that the hardware can either generate or verify the checksum.

Receiver Checksum Offload

When receive checksum offloading is enabled in the GEM_GXL, the IPv4 header checksum is checked as per RFC 791, where the packet meets the following criteria:

If present, the VLAN header must be four octets long and the CFI bit must not be set. (Also for receive one stacked VLAN is supported.)

- Encapsulation must be RFC 894 Ethernet Type Encoding or RFC 1042 SNAP Encoding or PPPoE Encoding.
- IPv4 packet.
- IP header is of a valid length.
- IP options are supported.

The GEM also checks the TCP checksum as per RFC 793, or the UDP checksum as per RFC 768, if the following criteria are met:

- IPv4 or IPv6 packet
- IP options and all IPv6 extension headers (i.e. hop-by-hop, routing and destination) are supported (except for fragmentation headers)
- Good IP header checksum (if IPv4).

- IP fragmentation is not supported (If a packet is fragmented, then the checksum will not be checked)
- Reserved bit must not be set in the IPv4 header flags field
- TCP or UDP packet.

When an IP, TCP or UDP frame is received, the receive buffer descriptor gives an indication if the GEM_GXL was able to verify the checksums. There is also an indication if the frame had SNAP encapsulation. These indication bits will replace the type ID match indication bits when the receive checksum offload is enabled. For details of these indication bits please refer to the table Receive Buffer Descriptor Entry

If any of the checksums are verified incorrect by the GEM_GXL, the packet is discarded and the appropriate statistics counter incremented.

Transmitter Checksum Offload

The transmitter checksum offload is only available if the GEM_GXL is configured to use the DMA in packet buffer mode and full store and forward mode is enabled. This is because the complete frame to be transmitted must be read into the packet buffer memory before the checksum can be calculated and written back into the headers at the beginning of the frame.

Transmitter checksum offload is enabled by setting bit [11] in the DMA Configuration Register. When enabled, it will monitor the frame as it is written into the transmitter packet buffer memory to automatically detect the protocol of the frame. Protocol support is identical to the receiver checksum offload.

For transmit checksum generation and substitution to occur, the protocol of the frame must be recognised and the frame must be provided without the FCS field, by making sure that bit [16] of the transmit descriptor word 1 is clear. If the frame data already had the FCS field, this would be corrupted by the substitution of the new checksum fields. Stacked VLANs are supported as long as the VLAN type field of the stacked VLAN is set to 88a8. This differs from receive where the stacked VLAN type field is a programmable value.

If these conditions are met, the transmit checksum offload engine will calculate the IP, TCP and UDP checksums as appropriate. Once the full packet is completely written into packet buffer memory, the checksums will be valid and the relevant DPRAM locations will be updated for the new checksum fields as per standard IP/TCP and UDP packet structures.

If the transmitter checksum engine is prevented from generating the relevant checksums, bits [22:20] of the transmitter DMA writeback status will be updated to identify the reason for the error. Note that the frame will still be transmitted but without the checksum substitution, as typically the reason that the substitution did not occur was that the protocol was not recognised.

MAC Filtering Block

The filter block determines which frames should be written to the external FIFO interface and on to the optional DMA.

Whether a frame is passed depends on what is enabled in the network configuration register, the state of the external matching pins, the contents of the specific address, type and hash registers and the frame's destination address and type field.

If bit 25 of the network configuration register is not set, a frame will not be copied to memory if the GEM_GXL is transmitting in half duplex mode at the time a destination address is received.

Ethernet frames are transmitted a byte at a time, least significant bit first. The first six bytes (48 bits) of an Ethernet frame make up the destination address. The first bit of the destination address, which is the LSB of the first byte of the frame, is the group or individual bit. This is one for multicast addresses and zero for unicast. The *all ones* address is the broadcast address and a special case of multicast.

The GEM_GXL supports recognition of specific source or destination addresses. The number of specific source or destination address filters is configurable and can range from zero to 36. Each specific address filter requires two registers, specific address register bottom and specific address register top. Specific address register bottom stores the first four bytes of the compared source or destination address. Specific address register top contains the last two bytes of this address, a control bit to select between source or destination address filtering and a 6-bit byte mask field to allow the user to mask bytes during the comparison. The first filter (Filter 1) is slightly different to all other filters in that there is no byte mask. Instead address comparison against individual bits of specific address register 1 can be masked using the unique specific address mask register. The addresses stored in all filters can be specific (unicast), group (multicast), local or universal.

The destination or source address of received frames is compared against the data stored in the specific address registers once they have been activated. The addresses are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. If a receive frame address matches an active address, the frame is written to the external FIFO interface and on to DMA memory if used.

Frames may be filtered using the type ID field for matching. Four type ID registers exist in the register address space and each can be enabled for matching by writing a one to the MSB (bit 31) of the respective register. When a frame is received, the matching is implemented as an OR function of the various types of match.

The contents of each type ID registers (when enabled) are compared against the length/type ID of the frame being received (e.g. bytes 13 and 14 in non-VLAN and non-SNAP encapsulated frames) and copied to memory if a match is found. The encoded type ID match bits (Word 0, Bit 22 and Bit 23) in the receive buffer descriptor status are set indicating which type ID register generated the match, if the receive checksum offload is disabled.

The reset state of the type ID registers is zero, hence each is initially disabled.

The following example illustrates the use of the address and type ID match registers for a MAC address of 21:43:65:87:A9:CB:

Preamble	55
SFD	D5
DA (Octet 0 - LSB)	21
DA (Octet 1)	43
DA (Octet 2)	65
DA (Octet 3)	87
DA (Octet 4)	A9
DA (Octet 5 - MSB)	CB
SA (LSB)	00*
SA	00*
SA	00*
SA	00*
SA	00*
SA (MSB)	00*
Type ID (MSB)	43
Type ID (LSB)	21

Note: ** - contains the address of the transmitting device.*

The sequence above shows the beginning of an Ethernet frame. Byte order of transmission is from top to bottom as shown. For a successful match to specific address 1, the following address matching registers must be set up:

Specific address 1 bottom (Address 0x088)	0x87654321
Specific address 1 top (Address 0x08C)	0x0000CBA9

And for a successful match to the type ID, the following type ID match 1 register must be set up:

Type ID Match 1 (Address 0x0A8) 0x80004321

Broadcast Address

Frames with the broadcast address of 0xFFFFFFFFFFFF are stored to memory only if the 'no broadcast' bit in the network configuration register is set to zero.

Hash Addressing

The hash address register is 64 bits long and takes up two locations in the memory map. The least significant bits are stored in hash register bottom and the most significant bits in hash register top.

The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames. The destination address is reduced to a 6 bit index into the 64 bit hash register using the following hash function. The hash function is an XOR of every sixth bit of the destination address.

```
hash_index[05] = da[05] ^ da[11] ^ da[17] ^ da[23] ^ da[29] ^ da[35] ^ da[41] ^ da[47]
hash_index[04] = da[04] ^ da[10] ^ da[16] ^ da[22] ^ da[28] ^ da[34] ^ da[40] ^ da[46]
hash_index[03] = da[03] ^ da[09] ^ da[15] ^ da[21] ^ da[27] ^ da[33] ^ da[39] ^ da[45]
hash_index[02] = da[02] ^ da[08] ^ da[14] ^ da[20] ^ da[26] ^ da[32] ^ da[38] ^ da[44]
hash_index[01] = da[01] ^ da[07] ^ da[13] ^ da[19] ^ da[25] ^ da[31] ^ da[37] ^ da[43]
hash_index[00] = da[00] ^ da[06] ^ da[12] ^ da[18] ^ da[24] ^ da[30] ^ da[36] ^ da[42]
```

`da[0]` represents the least significant bit of the first byte received, that is, the multicast/unicast indicator, and `da[47]` represents the most significant bit of the last byte received.

If the hash index points to a bit that is set in the hash register then the frame will be matched according to whether the frame is multicast or unicast.

A multicast match will be signalled if the multicast hash enable bit is set, `da[0]` is logic 1 and the hash index points to a bit set in the hash register.

A unicast match will be signalled if the unicast hash enable bit is set, `da[0]` is logic 0 and the hash index points to a bit set in the hash register.

To receive all multicast frames, the hash register should be set with all ones and the multicast hash enable bit should be set in the network configuration register.

External Address Matching

In applications where the internal address matching circuitry of the GEM_GXL is not sufficient, an external address matching interface is provided. Matches can be performed on source address, destination address, type or VLAN ID. Each field is provided as an output (`ext_sa`, `ext_da`, `ext_type` and `ext_vid` output pins) with a validating strobe signal (`ext_sa_stb`, `ext_da_stb`, `ext_type_stb` and `ext_vid_stb` output pins). The strobes for each field are asserted high for one cycle at the point the field becomes valid within the frame reception.

If receive checksum offload is enabled in the network control register, IP source and destination addresses are also available for external address matching along with UDP/TCP port numbers. These fields are provided as the `ext_ip_sa` and `ext_ip_da` outputs, and are validated by the `ext_ip_sa_stb` and `ext_ip_da_stb` outputs.

The external filter interface is enabled by setting bit 9 in the network configuration register.

External logic connected to the filter interface should return the `ext_match1`, `ext_match2`, `ext_match3` or `ext_match4` inputs when a match is found. Any one of the four match inputs can indicate a filter match as the inputs are logically ORed within the MAC.

The time at which the output strobes for the various fields are valid is fixed relative to the time of frame reception and depends on the type of frame being decoded. Destination address, source address and VLAN ID are always fixed relative to the start of the frame. Type ID and the IP address, however, depend on whether the frame has a VLAN tag and/or a valid SNAP encapsulation.

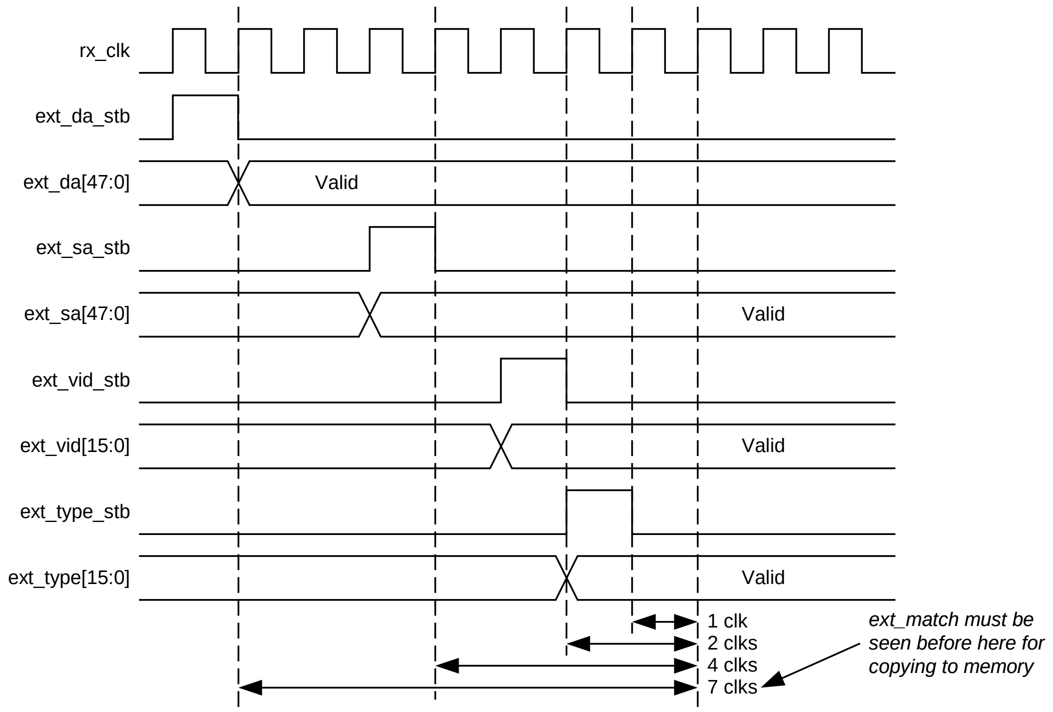
The number of `rx_clk` cycles available to make the comparison is design specific, depending on the parameterizable depth of the MAC receive pipeline. The `‘gem_rx_pipeline_delay` define in `gem_gxl_defs.v` defines the depth of the receive pipeline. A decision as to whether the frame should be stored to memory is made when the destination address reaches the end of the pipeline.

Timings from each strobe becoming valid until the internal match takes place are shown below for the default receive pipeline of ten.

These timings also demonstrate the worst case of TBI gigabit mode where 16 bits of data are processed every `rx_clk` cycle. For demonstration, the frame received has a VLAN tag. The type ID field, therefore, presented to the external filter interface is delayed until after the VLAN ID is presented so that the real type ID field is captured.

Strobe Signal	**No of rx_clk cycles for a match**
`ext_da_stb` (destination address)	7
`ext_sa_stb` (source address)	4
`ext_type_stb` (type field)	1
`ext_vid_stb` (VLAN ID)	2

The interface assumes the connecting signals are synchronous to `rx_clk` as shown in the following diagram.



The depth of the default receive pipeline is insufficient for external matching of the extracted IP addresses. To cope with both VLAN and SNAP frames, the pipeline must be at least 25 deep when matching on the extracted IP destination address. SNAP and IP frames are only identified if the receive checksum offload is enable in the network configuration register.

Copy All Frames (or Promiscuous Mode)

If the copy all frames bit is set in the network configuration register then all frames (except those that are too long, too short, have FCS errors or have `rx_er` asserted during reception) will be copied to memory. Frames with FCS errors will be copied if bit 26 is set in the network configuration register.

Disable Copy of Pause Frames

Pause frames can be prevented from being written to memory by setting the disable copying of pause frames control bit 23 in the network configuration register. When set, pause frames are not copied to memory regardless of the copy all frames bit, whether a hash match is found, a type ID match is identified or if a destination address match is found.

VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

TPID (Tag Protocol Identifier) 16 bits	**TCI (Tag Control Information) 16 bits**
0x8100	First 3 bits priority, then CFI bit, last 12 bits VID

The VLAN tag is inserted at the 13th byte of the frame adding an extra four bytes to the frame. To support these extra four bytes, the GEM can accept frame lengths up to 1536 bytes by setting bit 8 in the network configuration register.

If the VID (VLAN identifier) is null (0x000) this indicates a priority-tagged frame.

The following bits in the receive buffer descriptor status word give information about VLAN tagged frames:-

Bit 21 set if receive frame is VLAN tagged (i.e. type id of 0x8100).

Bit 20 set if receive frame is priority tagged (i.e. type id of 0x8100 and null VID). (If bit 20 is set bit 21 will be set also.).

Bit 19, 18 and 17 set to priority if bit 21 is set.

Bit 16 set to CFI if bit 21 is set.

The GEM_GXL can be configured to reject all frames except VLAN tagged frames by setting the discard non-VLAN frames bit in the network configuration register.

Wake on LAN Support

The receive block supports Wake on LAN by detecting the following events on incoming receive frames:

Magic packet

ARP request to the device IP address

Specific address 1 filter match

Multicast hash filter match

If one of these events occurs Wake on LAN detection is indicated by asserting the `wol` output pin for 64 `rx_clk` cycles. These events can be individually enabled through bits[19:16] of the Wake on LAN register. Also, for Wake on LAN detection to occur receive enable must be set in the network control register, however a receive buffer does not have to be available. In addition to the wol output pin, an optional wol interrupt is generated for software purposes. If only the mac receive clock is running at the time of the Wake on LAN event (and not the APB clock), the wol interrupt will not occur.

`wol` assertion due to ARP request, specific address 1 or multicast filter events will occur even if the frame is errored. For magic packet events, the frame must be correctly formed and error free.

A magic packet event is detected if all of the following are true:

magic packet events are enabled through bit 16 of the Wake on LAN register

the frame's destination address matches specific address 1

the frame is correctly formed with no errors

the frame contains at least 6 bytes of 0xFF for synchronization

there are 16 repetitions of the contents of specific address 1 register immediately following the synchronization

An ARP request event is detected if all of the following are true:

ARP request events are enabled through bit 17 of the Wake on LAN register

broadcasts are allowed by bit 5 in the network configuration register

the frame has a broadcast destination address (bytes 1 to 6)

the frame has a typeID field of 0x0806 (bytes 13 and 14)

the frame has an ARP operation field of 0x0001 (bytes 21 and 22)

the least significant 16 bits of the frame's ARP target protocol address (bytes 41 and 42) match the value programmed in bits[15:0] of the Wake on LAN register

The decoding of the ARP fields adjusts automatically if a VLAN tag is detected within the frame. The reserved value of 0x0000 for the Wake on LAN target address value will not cause an ARP request event, even if matched by the frame.

A specific address 1 filter match event will occur if all of the following are true:

specific address 1 events are enabled through bit 18 of the Wake on LAN register

the frame's destination address matches the value programmed in the specific address 1 registers

A multicast filter match event will occur if all of the following are true:

multicast hash events are enabled through bit 19 of the Wake on LAN register

multicast hash filtering is enabled through bit 6 of the network configuration register

the frame destination address matches against the multicast hash filter

the frame destination address is not a broadcast

IEEE 1588 and IEEE 802.1AS Support

IEEE 1588 is a standard for precision time synchronization in local area networks. It works with the exchange of special Precision Time Protocol (PTP) frames. The PTP messages can be transported over IEEE 802.3/Ethernet, over Internet Protocol Version 4 or over Internet Protocol Version 6 as described in the annex of IEEE Std 1588-2008.

Most 1588 functionality can be implemented in software but for greatest accuracy hardware assist is required to detect when PTP event messages pass the GMII interface (clock time-stamp point).

The GEM_GXL detects when the PTP event messages: sync, delay_req, pdelay_req and pdelay_resp are transmitted and received.

GEM_GXL output pins indicate the message time-stamp point (asserted on the start packet delimiter and de-asserted at end of frame) for all frames and the passage of PTP event frames (asserted when a PTP event frame is detected and de-asserted at end of frame).

Synchronization between master and slave clocks is a two stage process.

First the offset between the master and slave clocks is corrected by the master sending a sync frame to the slave with a follow up frame containing the exact time the sync frame was sent. Hardware assist modules at the master and slave side detect exactly when the sync frame was sent by the master and received by the slave. The slave then corrects its clock to match the master clock.

Second the transmission delay between the master and slave is corrected. The slave sends a delay request frame to the master which sends a delay response frame in reply. Hardware assist modules at the master and slave side detect exactly when the delay request frame was sent by the slave and received by the master. The slave will now have enough information to adjust its clock to account for delay. For example if the slave was assuming zero delay the actual delay will be half the difference between the transmit and receive time of the delay request frame (assuming equal transmit and receive times) because the slave clock will be lagging the master clock by the delay time already.

For hardware assist it is necessary to time-stamp when sync and delay_req messages are sent and received. The time-stamp is taken when the message time-stamp point passes the clock time-stamp point. For Ethernet the message time-stamp point is the SFD and the clock time-stamp point is the MII interface. (The 1588 spec refers to sync and delay_req messages as event messages as these require time-stamping. Follow up, delay response and management messages do not require time-stamping and are referred to as general messages.)

1588 version 2 defines two additional PTP event messages. These are the peer delay request (Pdelay_Req) and peer delay response (Pdelay_Resp) messages. These messages are used to calculate the delay on a link. Nodes at both ends of a link send both types of frames (regardless of whether they contain a master or slave clock). The Pdelay_Resp message contains the time at which a Pdelay_Req was received and is itself an event message. The time at which a Pdelay_Resp message is received is returned in a Pdelay_Resp_Follow_Up message.

1588 version 2 introduces transparent clocks of which there are two kinds, peer-to-peer (P2P) and end-to-end (E2E). Transparent clocks measure the transit time of event messages through a bridge and amend a correction field within the message to allow for the transit time. P2P transparent clocks additionally correct for the delay in the receive path of the link using the information gathered from the peer delay frames. With P2P transparent clocks delay_req messages are not used to measure link delay. This simplifies the protocol and makes larger systems more stable.

The sof_tx and sof_rx signals are provided to indicate the message time-stamp point and follow up signals are provided to indicate the presence of an event frame. With 1588 version 1 for a given data-rate the assertion of the event frame signals will be a fixed delay after the sof signals so taking the time-stamp could be delayed until the event signals are asserted and suitable compensation made.

The GEM_GXL recognizes ten different encapsulations for PTP event messages:

1. 1588 version 1 (UDP/IPv4 multicast)
2. 1588 version 1 (UDP/IPv4 multicast with VLAN)

3. 1588 version 2 (UDP/IPv4 multicast)
4. 1588 version 2 (UDP/IPv4 multicast with VLAN)
5. 1588 version 2 (UDP/IPv4 unicast)
6. 1588 version 2 (UDP/IPv4 unicast with VLAN)
7. 1588 version 2 (UDP/IPv6 multicast)
8. 1588 version 2 (UDP/IPv6 multicast with VLAN)
9. 1588 version 2 (Ethernet multicast)
10. 1588 version 2 (Ethernet multicast with VLAN)

Unicast PTP frame recognition is enabled via bit 20 of the Network Control Register.. The unicast addresses themselves are user programmable via the unicast PTP address register for which there are two (0x0D4 and 0x0D8). The first holds the RX unicast IP destination address and the other the TX unicast IP destination address. The unicast PTP address register should only be changed when the unicast PTP frame recognition is disabled.

Example of a sync frame in the 1588 version 1 format:

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	
SA (Octets 6 - 11)	
Type (Octets 12-13)	0800
IP header (Octets 14-22)	
IP header - UDP identifier (Octet 23)	11
IP header (Octets 24-29)	
IP header - DA (Octets 30-32)	E00001
IP header - DA (Octet 33)	81 or 82 or 83 or 84
UDP header - Source port (Octets 34-35)	
UDP header - Dest port (Octets 36-37)	013F
UDP header – length and checksum (Octets 38-41)	
PTP header (Octet 42)	
PTP header - versionPTP (Octet 43)	01
PTP header (Octets 44-73)	
PTP header - control (Octet 74)	00
PTP header (Octets 75-168)	

Example of a delay request frame in the 1588 version 1 format:

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	
SA (Octets 6 - 11)	
Type (Octets 12-13)	0800
IP header (Octets 14-22)	
IP header - UDP identifier (Octet 23)	11
IP header (Octets 24-29)	
IP header - DA (Octets 30-32)	E00001
IP header - DA (Octet 33)	81 or 82 or 83 or 84
UDP header - Source port (Octets 34-35)	
UDP header - Dest port (Octets 36-37)	013F
UDP header - length and checksum (Octets 38-41)	
PTP header (Octet 42)	
PTP header - versionPTP (Octet 43)	01
PTP header (Octets 44-73)	
PTP header - control (Octet 74)	01
PTP header (Octets 75-168)	

For 1588 version 1 messages sync and delay request frames are indicated by the GEM_GXL if the frames type field indicates TCP/IP, UDP protocol is indicated, the destination IP address is 224.0.1.129/130/131 or 132, the destination UDP port is 319 and the control field is correct.

The control field is 0x00 for sync frames and 0x01 for delay request frames.

For 1588 version 2 messages the type of frame is determined by looking at the message type field in the first byte of the PTP frame. Whether a frame is version 1 or version 2 can be determined by looking at the version PTP field in the second byte of both version 1 and version 2 PTP frames.

In version 2 messages sync frames have a message type value of 0x0, delay_req have 0x1, pdelay_req have 0x2 and pdelay_resp have 0x3.

Example of a sync frame in the 1588 version 2 (UDP/IPv4) format:

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	
SA (Octets 6 - 11)	
Type (Octets 12-13)	0800
IP header (Octets 14-22)	
IP header - UDP identifier (Octet 23)	11
IP header (Octets 24-29)	
IP header - DA (Octets 30-33)	E0000181
UDP header - source port (Octets 34-35)	
UDP header - dest port (Octets 36-37)	013F
UDP header - length and checksum (Octets 38-41)	
PTP header - messagetype (Octet 42[3:0])	0
PTP header - transport specific (Octet 42[7:4])	not checked
PTP header - version PTP (Octet 43)	02

Example of a pdelay_req frame in the 1588 version 2 (UDP/IPv4) format:

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	
SA (Octets 6 - 11)	
Type (Octets 12-13)	0800
IP header (Octets 14-22)	
IP header - UDP identifier (Octet 23)	11
IP header (Octets 24-29)	
IP header - DA (Octets 30-33)	E000006B
source IP port (Octets 34-35)	
dest IP port (Octets 36-37)	013F
UDP header - other stuff (Octets 38-41)	
PTP header - messagetype (Octet 42[3:0])	2
PTP header - transport specific (Octet 42[7:4])	not checked
PTP header - version PTP (Octet 43)	02

Example of a sync frame in the 1588 version 2 (UDP/IPv6) format:

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	
SA (Octets 6 - 11)	
Type (Octets 12-13)	86dd
IP header (Octets 14-19)	
IP header - UDP identifier (Octet 20)	11
IP header (Octets 21-37)	
IP header - DA (Octets 38-53)	FF0X000000000181
UDP header - source port (Octets 54-55)	

UDP header - dest port (Octets 56-57)	013F
UDP header - (Octets 58-61)	
PTP header - messagetype (Octet 62[3:0])	0
PTP header - transport specific (Octet 62[7:4])	not checked
PTP header - version PTP (Octet 63)	02

Example of a pdelay_resp frame in the 1588 version 2 (UDP/IPv6) format:

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	
SA (Octets 6 - 11)	
Type (Octets 12-13)	86dd
IP header (Octets 14-19)	
IP header - UDP identifier (Octet 20)	11
IP header (Octets 21-37)	
IP header - DA (Octets 38-53)	FF0200000000006B
UDP header - source port (Octets 54-55)	
UDP header - dest port (Octets 56-57)	013F
UDP header - (Octets 58-61)	
PTP header - messagetype (Octet 62[3:0]	3
PTP header - transport specific (Octet 62[7:4])	not checked
PTP header - version PTP (Octet 63)	02

Example of a sync frame in the 1588 version 2 (Ethernet multicast) format. For the multicast address 011B19000000 sync and delay request frames are recognized depending on the messagetype field, 00 for sync and 01 for delay request:

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	011B19000000
SA (Octets 6 - 11)	
Type (Octets 12-13)	88F7
messagetype (Octet 14[3:0])	0
transport specific (Octet 14[7:4])	not checked
version PTP (Octet 15)	02

Example of a pdelay_req frame in the 1588 version 2 (Ethernet multicast) format which uses the “nearest bridge group address” of 0180C200000E as defined in 802.1Q. 0180C200000E is a special multicast address that is not forwarded by bridges. Also 802.1AS requires the use of this address with Ethernet encapsulation without VLAN tagging (see 11.3.3 and 11.3.4 in the 802.1AS spec), so sync frames are recognized with this address in addition to pdelay request and pdelay response frames depending on the messagetype field, 00 for sync, 02 for pdelay request and 03 for pdelay response. However sync frames will not be indicated with this address if they are VLAN tagged.

Preamble/SFD	55555555555555D5
DA (Octets 0 - 5)	0180C200000E
SA (Octets 6 - 11)	
Type (Octets 12-13)	88F7
messagetype (Octet 14[3:0])	2
transport specific (Octet 14[7:4])	not checked
version PTP (Octet 15)	02

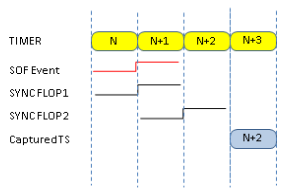
The GEM_GXL contains an optional time stamp unit (TSU) selectable with a tick define. The TSU consists of a timer and registers to capture the time at which PTP event frames cross the message time-stamp point. These are accessible through the GEM_GXL’s APB interface. An interrupt is issued when a capture register is updated.

Support for Timestamping and Timestamp Accuracy

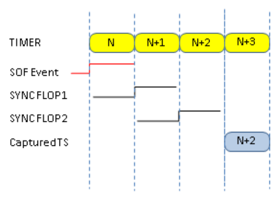
The MAC has the responsibility of sampling the TSU timer value when the TX or RX SOF event of the frame passes the MII/GMII boundary. This event is an existing signal synchronous to MAC TX/RX clock domains. The MAC uses the sampled timestamp to insert the timestamp into transmitted PTP sync frames (if one step sync feature is enabled), or to pass to the register block to capture the timestamp in APB accessible registers, or to pass to the DMA to insert into TX or RX descriptors. For each of these, the SOF event, which is captured in the tx_clk and rx_clk domains respectively, is synchronized to the tsu_clk domain and the resulting signal is used to sample the TSU count value. This value will be kept stable for an entire frame, or specifically at least 64 TX/RX clock cycles, since the minimum frame size in Ethernet is 64bytes and worst case is a transfer rate of 1byte per cycle. It is used as the source for all the various components within the GEM_GXL that require the timestamp value.

Since the SOF event had to pass a clock boundary, there is a degree of inaccuracy in the captured timestamp. The level of inaccuracy depends on the frequency of tsu_clk. There will be no more than 1 cycle of inaccuracy as shown below :

Timestamp Capture Best Case



Timestamp Capture Worst Case



In the best case, the SOF event (which is in the rx/tx clk domain) just meets the setup time of the tsu_clk domain at the input to the first sync flop. The captured TS is N+2, but it really should be N+1. In the worst case, the captured TS is also N+2, but it really should be N. There is 1 cycle of uncertainty.

Single Step Timestamping:

Support of one step clock for TX Sync frames can be enabled by setting a bit in the network control register. In this mode the timestamp field, within the 1588 (ver 2) sync frame, is replaced by the TSU timestamp value at the time the Sync frame sof passes the MII interface. To use single step timestamping, the sampled timestamp must be stable before the point at which the GEM_GXL requires to insert the timestamp. This can be guaranteed by enforcing a rule that TSU clock (tsu_clk) is greater than 1/8th the frequency of tx_clk or rx_clk. The UDP checksum offload functionality cannot be used with single step timestamping.

Single step update of correction field in PTP sync frames

Version 2 PTP frames have a common message header of 34 octets defined in Table 18 of the 1588-2008 spec. Sync frames contain the origin Timestamp immediately after the header. The correction field begins at the ninth octet in the header.

The correction field is defined in 13.3.2.7 of IEEE Std 1588-2008. The correction field is measured in nanoseconds and is multiplied by 2^16. For example 2.5ns is represented by the 64 bit value 0x0000000000028000 and is transmitted most significant byte first. A value of one in all bits, except the most significant, indicates that the correction is too big to be represented.

From release 1p09 onwards if bit 27 “oss_correction_field” is set in the network_control register then the correction field will be updated.

Section 11.5 of the 1588 spec describes updating of the correction field. Sections 13.3 and 13.6 describe the fields of the sync frame. At offset 8 in the PTP sync frame there is an 8 byte correction field. The requirement is to add the sync frame residence time to this field in a single step process.

The residence time is the difference between the ingress and egress timestamps of the sync frame. The ingress time stamp will be available to firmware as is the correction field which can be extracted from the received sync frame.

The TSU value is stored as nanoseconds and seconds, while the correction field is stored as nanoseconds (48 bits for ns and 16 bits for subns). The subns bits of the correction field can be used by firmware to signal to the gem_tx block what action it should take as timestamps cannot be taken with sub-nanosecond precision. Firmware can also update the correction field to take account of the ingress timestamp.

For single step updating of the correction field to work firmware has to do the following

1. Clear the 16 least significant bits of the correction field except any bits that need to be set as below
2. If the correction field is at its maximum value set bit[15] of the correction field to 1 and take no further action
3. Set bit[8] of the correction field to the value of the LSB of the captured ingress seconds TSU value. The captured value refers to the time the sync frame entered the SoC.
4. Subtract the nanosecond value of the ingress time-stamp from the correction field (ignoring the subns bits of the correction field). If the resulting value is negative set bit[14] of the correction field.
5. Write the absolute result (ie positive integer nanosecond number) to the correction field

Because firmware will have initialised the correction field by subtracting the nanosecond value of the ingress time-stamp from it, adding the nanosecond egress time to the correction field results in it being incremented by the residence time of the sync frame in device.

The GEM_GXL adds the egress nanosecond value to the correction field and clears the least significant 16 bits. If the egress least significant bit of the seconds value is different from the value stored in bit [8] then an extra billion nanoseconds are added to the correction field.

If there is an overflow or bit[15] is set then the correction field is set to the maximum value of 0x7FFFFFFFFFFFFFFF.

The least significant bits of the correction field are subnanoseconds and the GEM_GXL cannot take timestamps with this precision so it is OK to re-use the subnanosecond field of the correction field to carry seconds and other data. It is also assumed that the residence time of the sync frame will be less than a second so only a single second bit needs to be carried.

For reference for 802.1AS the originTimestamp field is reserved and the correction field is zero in sync frames because 802.1AS does not support one-step (single step) time stamping (see 7.5 f, 10.5.2.2.7 and 11.4.3 in the 802.1AS spec).

Timestamp Capture in APB registers

There are eight 80bit status registers that capture the time at which PTP event frames are transmitted and received. An interrupt is issued when these registers are updated.

Timestamp Capture in DMA descriptors

The TX and/or RX Timestamp can optionally be captured in an extended buffer descriptor when configured using bits in the DMA configuration register. The timestamp can be captured for a number of frame types (ptp event or ptp general, or all frames, or none as defined in tx/rx_bd_control registers) and a bit within BD word 0/1 is used to indicate that the timestamp is present.

Controlling the GEM_GXL TSU

The timer is implemented as a 102 bit register with the upper 48 bits counting seconds, the next 30 bits counting nanoseconds and the lowest 24 bits counting sub-nanoseconds. The lower 54 bits roll over when they have counted to one second. An interrupt is generated when the seconds increment. The timer value can be read, written and adjusted through the APB interface.
The timer is clocked by either pclk or by tsu_clk if gem_tsu_clk is defined the gem_gxl_defs.v file.

There are three modes of operation to control the way the timer varies over time. These are;

1) Increment timer by a fixed value every input clock (tsu_clk or pclk). This is increment mode.
2) Increment timer by a fixed value for a fixed number of input clocks, followed by an alternative increment value for a single clock. This is alternative increment mode. This is a legacy mode of operation and it is not recommended to use the alternative increment mode of operation.
3) Adjust the timer by adding or subtracting 1ns from the timer. This mode is controlled by gem_tsu_inc_ctrl and gem_tsu_ms inputs. This is timer adjust mode.

These modes are described below:

Increment Mode:

The amount by which the timer increments each clock cycle is controlled by the timer increment register. Bits 7:0 are the default increment value in nanoseconds and an additional 24 bits of sub-nsec resolution are available using the timer increment sub_nsec register. If the rest of the timer increment register is written with zero the timer increments by the value in 7:0 , plus sub-nsec register, each clock cycle.

The sub-nsec register allows a resolution of approximately 58.6 attoseconds (5.86E-17). The sub-nsec register is split into two fields with the MSBs at 15:0 and the LSBs at 31:24. This is because when the number of significant bits was increased from sixteen to twenty four the location of the MSBs was left unchanged and the LSBs were added to the top of the register. To calculate the value to write to the sub-ns register you take the decimal value of the sub-nanosecond value, then multiply it by by 2 to the power of 24 (16777216) and convert the result to hexadecimal. For example for a sub-nanosecond value of 0.33333333 you would write 0x55005555.

Alternative Increment Mode:

It is recommended not to use this legacy mode of operation.

Bits 15:8 of the increment register are the alternative increment value in nanoseconds and bits 23:16 are the number of increments after which the alternative increment value is used. If 23:16 are zero then the alternative increment value will never be used.

Taking the example of 10.2MHz you have 102 cycles every ten microseconds or 51 every five microseconds. So a timer with a 10.2MHz clock source is constructed by incrementing by 98ns for fifty cycles and then incrementing by 100ns (98*50+100=5000). This is programmed by setting the 1588 timer increment register to 0x00326462.

For a 49.8MHz clock source it would be 20ns for 248 cycles followed by an increment of 40ns (20*248+40=5000) programmed as 0x00F82814.

Having eight bits for the 'number of increments' field allows frequencies up to 50MHz to be supported with 200kHz resolution.

Timer Adjust Mode:

An alternative way of controlling the way the timer increment register is to use the gem_tsu_inc_ctrl inputs. When gem_tsu_inc_ctrl [1:0] =

- ✓ 2b'11 timer register increments as normal.
- ✓ 2b'01 timer register increments by an additional nanosecond.
- ✓ 2b'10 timer increments by a nanosecond less.
- ✓ 2b'00
  - o When gem_tsu_ms = 1b'1, the "nanoseconds" timer register is cleared and the "seconds" timer register is incremented with each clock cycle.
  - o When gem_tsu_ms = 1b'0, the timer register increments as normal but the timer value is copied to the sync strobe register.
  - o

The timer sync strobe register is loaded with the value of the timer when the input signals gem_tsu_ms and gem_tsu_inc_ctrl are set to zero.

The TSU timer count value bits are passed out of the core so that it can optionally be used directly by external hardware.

The TSU timer count value can be compared to a programmable comparison value. For the comparison the 48 bits of the seconds value and the upper 22 bits of the nanoseconds value are used. A signal is output from the core to indicate when the TSU timer count value is equal to the comparison value stored in the TSU timer comparison value registers (0x0DC, 0x0E0 and 0x0E4). An interrupt can also be generated (if enabled) when the TSU timer count value and comparison value are equal, mapped to bit 29 of the interrupt status register.

IEEE 802.1AS is mostly a subset of 1588. There is one difference in that 802.1AS uses the Ethernet multicast address 0180C200000E for sync frame recognition whereas 1588 does not. GEM_GXL is designed to recognize sync frames with both 802.1AS and 1588 addresses and so can support both 1588 and 802.1AS frame recognition simultaneously.

An optional external time stamp port can be used instead of the internal timer. The external port is 94 bits wide, conforming to the 94 most significant bits of the internal TSU count as defined above.

The external port must be synchronous to the supplied tsu_clk.

The external port is enabled using a bit in the Network Control Register.

When using external timestamp port the registers which modify the timestamp in internal mode, will have no effect, but the timer read and compare registers operate as described above.

MAC 802.3 Pause Frame Support

Note: See Clause 31, and Annex 31A and 31B of the IEEE standard 802.3 for a full description of pause operation.

The start of an 802.3 pause frame looks like this:

Destination Address	Source Address	Type (MAC Control Frame)	Pause Opcode	Pause Time
0x0180C2000001	6 bytes	0x8808	0x0001	2 bytes

The GEM_GXL supports both hardware controlled pause of the transmitter upon reception of a pause frame and hardware generated pause frame transmission.

802.3 Pause Frame Reception

Bit 13 of the network configuration register is the pause enable control for reception. If this bit is set transmission will pause if a non zero pause quantum frame is received.

If a valid pause frame is received then the pause time register is updated with the new frame's pause time regardless of whether a previous pause frame is active or not. An interrupt (either bit 12 or bit 13 of the interrupt status register) is triggered when a pause frame is received, but only if the interrupt has been enabled (bit 12 and bit 13 of the interrupt mask register). Pause frames received with non zero quantum are indicated through the interrupt bit 12 of the interrupt status register. Pause frames received with zero quantum are indicated on bit 13 of the interrupt status register.

Once the pause time register is loaded and the frame currently being transmitted has been sent, no new frames are transmitted until the pause time reaches zero. The loading of a new pause time, and hence the pausing of transmission, only occurs when the GEM_GXL is configured for full duplex operation. If the GEM_GXL is configured for half duplex there will be no transmission pause, but the pause frame received interrupt will still be triggered. A valid pause frame is defined as having a destination address that matches either the address stored in specific address register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0001.

Pause frames that have FCS or other errors will be treated as invalid and will be discarded. 802.3 Pause frames that are received after Priority based flow control (PFC) has been negotiated will also be discarded. Valid pause frames received will increment the pause frames received statistic register.

The pause time register decrements every 512 bit times once transmission has stopped. For test purposes, the retry test bit can be set (bit 12 in the network configuration register) which causes the pause time register to decrement every `tx_clk` cycle once transmission has stopped.

The interrupt (bit 13 in the interrupt status register) is asserted whenever the pause time register decrements to zero (assuming it has been enabled by bit 13 in the interrupt mask register). This interrupt is also set when a zero quantum pause frame is received.

802.3 Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit pause frame bits of the network control register and from the external input pins `tx_pause`, `tx_pause_zero` and `tx_pfc_sel`. If either bit 11 or bit 12 of the network control register is written with logic 1, or if the input signal `tx_pause` is toggled when `tx_pfc_sel` is low, an 802.3 pause frame will be transmitted providing full duplex is selected in the network configuration register and the transmit block is enabled in the network control register.

Pause frame transmission will happen immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise of the following:

A destination address of 01-80-C2-00-00-01

A source address taken from specific address register 1

A type ID of 88-08 (MAC control frame)

A pause opcode of 00-01

A pause quantum register

Fill of 00 to take the frame to minimum frame length

Valid FCS

The pause quantum used in the generated frame will depend on the trigger source for the frame as follows:

If bit 11 is written with a one, the pause quantum will be taken from the transmit pause quantum register. The transmit pause quantum register resets to a value of 0xFFFF giving maximum pause quantum as default.

If bit 12 is written with a one, the pause quantum will be zero.

If the `tx_pause` input is toggled, `tx_pfc_sel` is low and the `tx_pause_zero` input is held low until the next toggle, the pause quantum will be taken from the transmit pause quantum register.

If the `tx_pause` input is toggled, `tx_pfc_sel` is low and the `tx_pause_zero` input is held high until the next toggle, the pause quantum will be zero.

After transmission, a pause frame transmitted interrupt will be generated (bit 14 of the interrupt status register) and the only statistics register that will be incremented will be the pause frames transmitted register.

Pause frames can also be transmitted by the MAC using normal frame transmission methods.

MAC PFC Priority Based Pause Frame Support

Note: Refer to the 802.1Qbb standard for a full description of priority based pause operation.

The GEM_GXL supports PFC Priority Based Pause transmission and reception. Before PFC pause frames can be received, bit 16 of the network control register must be set.

The start of a PFC pause frame looks like this:

Destination Address	**Source Address**	**Type (MAC Control Frame)**	**Pause Opcode**	**Priority Enable Vector**	**Pause Times**
0x0180C2000001	6 bytes	0x8808	0x0101	2 bytes	8 * 2 bytes

PFC Pause Frame Reception

The ability to receive and decode priority based pause frames is enabled by setting bit 16 of the network control register. When this bit is set, the GEM_GXL will match either classic 802.3 pause frames or PFC priority based pause frames. Once a priority based pause frame has been received and matched, then from that moment on the GEM_GXL will only match on priority based pause frames (this is an 802.1Qbb requirement, known as PFC negotiation). Once priority based pause has been negotiated, any received 802.3x format pause frames will not be acted upon. The state of PFC negotiation is identified via the output `pfc_negotiate`.

If a valid priority based pause frame is received then the GEM_GXL will decode the frame and determine which, if any, of the 8 priorities require to be paused. Up to 8 pause time registers are then updated with the 8 pause times extracted from the frame regardless of whether a previous pause operation is active or not. An interrupt (bit 12 of the interrupt status register) is triggered when a non-zero PFC pause frame is received, but only if the interrupt has been enabled (bit 12 of the interrupt mask register). Pause frames received with non-zero quantum are indicated through the interrupt bit 12 of the interrupt status register. PFC pause frames received with zero quantum cannot trigger an interrupt; that is bit 13 is never set for PFC pause frames. The output vector `rx_pfc_paused` is used to indicate when the pause time for a particular priority reaches zero. The state of the 8 pause time counters are indicated through the outputs `rx_pfc_paused.` These outputs will remain high for the duration of the pause time quanta. The loading of a new pause time only occurs when the GEM_GXL is configured for full duplex operation. If the GEM_GXL is configured for half duplex, the pause time counters will not be loaded, but the pause frame received interrupt will still be triggered. A valid pause frame is defined as having a destination address that matches either the address stored in specific address register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0101.

Pause frames that have FCS or other errors will be treated as invalid and will be discarded. Valid pause frames received will increment the pause frames received statistic register.

The pause time registers decrement every 512 bit times immediately following the PFC frame reception. For test purposes, the retry test bit can be set (bit 12 in the network configuration register) which causes the pause time register to decrement every `rx_clk` cycle once transmission has stopped.

PFC Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit priority based pause frame bit of the network control register and from the external input pins `tx_pause`, `tx_pfc_pause[7:0], tx_pfc_pause_zero[7:0]` and `tx_pfc_sel`. If bit 17 of the network control register is written with logic 1, or if the input signal `tx_pause` is toggled when `tx_pfc_sel` is high, a PFC pause frame will be transmitted providing full duplex is selected in the network configuration register and the transmit block is enabled in the network control register. When bit 17 of the network control register is set, the fields of the priority based pause frame will be built using the values stored in the transmit pfc pause register.

Pause frame transmission will happen immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise of the following:

A destination address of 01-80-C2-00-00-01

A source address taken from specific address register 1

A type ID of 88-08 (MAC control frame)

A pause opcode of 01-01

A priority enable vector taken from `tx_pfc_pause` or the transmit pfc pause register

8 pause quantum registers

Fill of 00 to take the frame to minimum frame length

Valid FCS

The pause quantum registers used in the generated frame will depend on the trigger source for the frame as follows:

If bit 17 of the network control register is written with a one then the priority enable vector of the priority based pause frame will be set equal to the value stored in the transmit pfc pause quantum register [7:0]. For each entry equal to zero in the transmit pfc pause quantum register[15:8], the pause quantum field of the pause frame associated with that entry will be taken from the transmit pause quantum register. For each entry equal to one in the transmit pfc pause quantum register[15:8], the pause quantum associated with that entry will be zero.

The transmit pause quantum register resets to a value of 0xFFFF giving maximum pause quantum as default.

The pause quantum registers are classed as static and these registers should be updated only when no pfc frame is being transmitted.

A configuration option to use 8 different pause quantums, one for each pause priority, is available by defining gem_pfc_multi_quantum in the gem_defs_*.v file.

When this define is active, the default operation is to use bits[15:0] of the original transmit pfc pause quantum register for every pause priority.

To use the eight pause priority quantum register, set Network Control Register bit[24] = '1'. This will cause tx_pause_quantum/1/2/3 register to be used, (note that two priority quantums are stored per register, one in bits [31:16] and one in bits [15:0]).

If the `tx_pause` input is toggled and `tx_pfc_sel` is high then the priority enable vector of the priority based pause frame will be set equal to the value in `tx_pfc_pause` [7:0]. For each entry equal to zero in `tx_pfc_pause_zero`[7:0], the pause quantum field of the pause frame associated with that entry will be taken from the transmit pause quantum register. For each entry equal to one in `tx_pfc_pause_zero` [7:0], the pause quantum associated with that entry will be zero.

After transmission, a pause frame transmitted interrupt will be generated (bit 14 of the interrupt status register) and the only statistics register that will be incremented will be the pause frames transmitted register.

PFC Pause frames can also be transmitted by the MAC using normal frame transmission methods.

Energy Efficient Ethernet support

IEEE 802.3az adds support for energy efficiency to Ethernet. These are the key features of 802.3az:

- Allows a system's transmit path to enter a low power mode if there is nothing to transmit.
- Allows a PHY to detect whether its link partner's transmit path is in low power mode, therefore allowing the system's receive path to enter low power mode.
- Link remains up during lower power mode and no frames are dropped.
- Asymmetric, one direction can be in low power mode while the other is transmitting normally.
- LPI (Low Power Idle) signaling is used to control entry and exit to and from low power modes.
- LPI signaling can only take place if both sides have indicated support for it through auto-negotiation.
-

802.3az operation:

- Low power control is done at the MII (reconciliation sublayer).
- As an architectural convenience in writing the 802.3az it is assumed that transmission is deferred by asserting carrier sense, in practice it will not be done this way. This system will know when it has nothing to transmit and only enter low power mode when it is not transmitting.
- LPI should not be requested unless the link has been up for at least one second.
- LPI is signaled on the GMII transmit path by asserting 0x01 on txd with tx_en low and tx_er high.
- A PHY on seeing LPI requested on the MII will send the sleep signal before going quiet. After going quiet it will periodically emit refresh signals.
- The sleep, quiet and refresh periods are defined in Table 78-2 of 802.3az. For 1000BASE-X the sleep period is 20 microseconds, the quiet period 2.5 milliseconds and the refresh period 20 microseconds.

- 1000BASE-X is required to go quiet after sleep is signaled. The easiest way to do this is to write to a control register to disable transmit in the SerDes.
- SGMII and XFI are not part of 802.3az and should not go quiet after sleep is signaled.
- LPI mode ends by transmitting normal idle for the wake time. There is a default time for this but it can be adjusted in software using the Link Layer Discovery Protocol (LLDP) described in Clause 79 of 802.3az.
- LPI is indicated at the receive side when sleep and refresh signaling has been detected.

LPI Operation in Cadence IP

It is best to use firmware to control LPI. LPI operation is something that happens at the system level. Firmware gives maximum control and flexibility of operation. LPI operation is straightforward and firmware should be capable of responding within the required timeframes.

Autonegotiation

- Indicate EEE capability using next page autonegotiation
-

For the transmit path:

- If the link has been up for 1 second and there is nothing being transmitted write to the LPI bit in network control register.
- If connected to 1000BASE-T PHY using SGMII or RGMII there is nothing more to do.
- If connected to a backplane using a 1000BASE-KX PHY use firmware to periodically disable the SerDes transmit path. (Write to bit 1.160.0 for 1000BASE-KX.)
- Wake up by clearing the LPI bit in the network control register.
-

From the gem_gxl 1p07 release onwards the LPI bit is ORed into the transmit pause functionality so transmission will pause as long as tx_lpi_en is active. This allows queuing of new transmit frames while the LPI signal is active. The 16 bit Tw_sys_tx time register at 0x060 holds the system wake time. After tx_lpi_en is de-asserted transmission will continue to be paused while the value in the tw_sys_tx_time counts done fixed time periods. The time period counted down will be 64ns for gigabit operation, 320 ns for 100M operation and 3200 ns for 10M operation. (In other words 8 tx_clk periods). So for example if tw_sys_tx_time is set to 8 in 100M mode transmission will be paused for 8 * 320 ns = 25.6 microseconds after deassertion of tx_lpi_en.

For the receive path:

- Wait for an interrupt to indicate that LPI has been received.
- Disable relevant parts of the receive path if desired but keep the PCS and SerDes active.
- Wait for an interrupt to indicate that regular idle has been received and then re-enable the receive path.

PHY Interface

The following PHY interfaces are supported by the Gigabit Ethernet MAC:

GMII

MII

RMII

RGMII

TBI

The PCS select bit in the network configuration register selects between the TBI and the GMII/MII. The GMII should only be used for 1000 Mbps operation. The TBI interface may be used for 1000BASE-X operation or SGMII operation. The MII interface is provided for 10/100 operation and uses `txd[3:0]` and `rxd[3:0]`, with `txd[7:4]` and `rxd[7:4]` not used. The unused `rxd[7:4]` inputs must be tied to either logic zero or logic one, when MII interface is being used.

10/100/1000 Operation

The gigabit select bit in the network configuration register selects between 10/100 Mbps Ethernet operation and 1000 Mbps mode. The 10/100 Mbps speed bit in the network configuration register is used to select between 10 Mbps and 100 Mbps, with the only affect being a value change in the `speed_mode[2:0]` output.

SGMII Operation

The PCS may be configured to operate in SGMII mode. This allows the GEM to be used as a building block to support an SGMII interface to an external PHY further reducing the pin count.

When bit 27 (SGMII mode) in the network configuration register is set the PCS auto-negotiation advertisement and link partner ability registers have a different definition. Additionally, the time duration of the link timer is reduced from 10 ms to 1.6 ms.

Auto-negotiation is something that occurs between PHYs. SGMII is a MAC-PHY interconnect and the auto-negotiation functionality (defined in Clause 37 of IEEE 802.3) is used to transfer status information from the PHY to the MAC rather than to perform auto-negotiation.

In SGMII mode bits[11:10] of the link partner ability register return the data transfer rate of the link which will have been previously negotiated by the PHY with its link partner PHY). This could be gigabit, 100M or 10M. This information is used by configuration software to set bits 10 and 0 of the network configuration register.

The MAC transmit and receive data paths are reconfigured by the network configuration register bits 11 and 10 (PCS select and gigabit) for different modes and speeds of operation.

PCS select	Gigabit	Function
1	1	1000Mb/s using TBI interface
0	1	1000Mb/s using GMII interface
0	0	10/100Mb/s using MII interface
1	0	10/100Mb/s using TBI interface in SGMII mode

For a complete SGMII solution a SerDes hard macro is also required. This performs the serialisation and deserialisation of the data and translation between the 125/312.5MHz and 1.25/3.125GHz clock domains. A SerDes can also be used for SGMII transmit clock generation by providing it with parallel data of 10b’1010101010.

Jumbo Frames

The jumbo frames enable bit in the network configuration register allows the GEM_GXL in its default configuration to receive jumbo frames up to a software configurable number of bytes in size. This operation does not form part of the IEEE 802.3 specification and is normally disabled. When jumbo frames are enabled, frames received with a frame size greater than the configured value are discarded.

The default limit applied to jumbo frame lengths can be modified by changing the value of the `` `gem_jumbo_max_length `` configuration option, which defaults to 10,240 bytes.

The jumbo frames max length can also be controlled using the "Maximum Jumbo Frame Size" register.

The power on value of the "Maximum Jumbo Frame Size" register defaults to the `` `gem_jumbo_max_length `` define value

Note: The max_jumbo_frame_size register must be set to greater than 1536 when network_config[8] (receive_1536_byte_frames control bit) is set to '1',

Physical Coding Sub-Layer

The PCS sub-layer provides a TBI for connection to a PHY transceiver. It is operational when the PCS interface is selected through the PCS select bit in the network configuration register. Components of the PCS sub-system include a PCS transmitter, 8B/10B encoder, PCS receiver, 8B/10B decoder and auto-negotiation.

PCS Transmit

Ethernet transmit data is passed to the PCS transmitter in byte form from the MAC transmitter and encoded into 10-bit words by the 8B/10B encoder for transmission. The 8B/10B encoder module is controlled by the transmit state machine and generates both data code groups and special code groups as per the IEEE 802.3 standard.

The transmitter generates ordered sets consisting of both special code groups and data code groups. Each Ethernet frame is encapsulated between a start of frame and end of frame. If auto-negotiation is requested then the transmitter generates ordered sets to reconfigure the link.

When the PCS is not performing auto-negotiation and the MAC has no frame to transmit, the PCS transmitter generates idle code groups. Idle code groups will also be transmitted if the receive link is down unless bit 31 is set in the network configuration register.

Each MAC frame to be transmitted is encoded a byte at a time, with each 10 bit output word being based on the Ethernet octet for transmission and the running disparity from the previously transmitted word.

The PCS transmits its ten-bit symbols as alternate even and odd numbered code groups. The start packet delimiter has to be transmitted as an even numbered code group. To accomplish this the GEM may need to delete a byte of preamble, reducing the transmitted preamble from seven bytes to six. This is accordance with the 802.3 spec. Other implementations may not delete preamble and instead adjust the IPG so that it falls below 12 bytes.

For reference see the following sections of the 802.3 standard:

"36.2.4.14 Start_of_Packet (SPD) delimiter. A Start_of_Packet delimiter (SPD) is used to delineate the starting boundary of a data transmission sequence and to authenticate carrier events. Upon each fresh assertion of TX_EN by the GMII, and subsequent to the completion of PCS transmission of the current ordered_set, the PCS replaces the current octet of the MAC preamble with SPD."

If the PCS is configured to use a 20-bit interface to the PHY, an optional 10 to 20-bit gearbox is instantiated to perform the datawidth conversion and additional clock inputs will be present.

PCS Receiver

The receive interface between the PHY and the PCS is configurable and different clocks will be present depending on the options chosen.

If a 10-bit PHY interface is chosen, then an optional 10-20-bit gearbox is automatically instantiated to perform the datawidth conversion with additional clock inputs added.

If the interface selection is not set to gem_pcs_legacy_if then additional comma and code group alignment functions will be instantiated to ensure that the receive data is correctly aligned ready for decoding.

The internal datapath of the PCS is 16-bits for easier timing closure.

Two 8B/10B decoder blocks perform the decoding function for the data received via the TBI. Running disparity is performed on the received data, with disparity indicators being passed between the two decoders.

Each 8B/10B decoder decodes special code groups and data code groups as per IEEE 802.3 standard. A receive state machine monitors special code groups and determines whether the TBI contains Ethernet frames, auto-negotiation link configuration information or if the link is idle.

Decoded 8-bit data, along with `rx_dv` and `rx_er`, is generated and passed to the MAC receiver in 16-bit form.

For reference see the following sections of the 802.3 standard:

"35.2.3.2.2 Receive case. The conditions for assertion of RX_DV are defined in 35.2.2.7. The operation of 1000 Mb/s PHYs can result in shrinkage of the preamble between transmission at the source GMII and reception at the destination GMII. Table 35–3 depicts the case where no preamble bytes are conveyed across the GMII. This case may not be possible with a specific PHY, but illustrates the minimum preamble with which MAC shall be able to operate."

PCS Auto-negotiation

An auto-negotiation block provides a means for the PCS to establish automatic link configuration. It is performed at power-up or during normal operation if requested by a link partner or through the restart auto-negotiation bit in the PCS control register.

By default the GEM has auto-negotiation enabled in the PCS control register and full and half duplex capability enabled in the PCS auto-negotiation advertisement register. Pause capability is disabled in the advertisement register by default. If auto-negotiation is not required then bit 12 of the PCS control register needs to be set low.

When a new base or next page is received from the link partner a PCS link partner page received interrupt is set (bit 17 of the interrupt status register). The first time this interrupt is received, it will indicate a base page received and subsequent reads will indicate next pages.

For next page exchange to work the next page register (0x21c) must be written within 10 ms of receiving a new page from the link partner. If the link partner is requesting next pages and the GEM has none to send then the next page register should be written with the null message (0x2001). The value 0x0000 must not be written to the next page register.

The GEM signals completion of auto-negotiation through the PCS auto-negotiation complete interrupt, on bit 16 of the interrupt status register. Auto-negotiation completion is also indicated by bit 5 of the PCS status register.

The PCS resolves the GEM and link partner's abilities and reports the result in the network status register. Pause transmit and receive resolution are reported in accordance with table 37-4 in the IEEE 802.3 specification. If full duplex capability is resolved the duplex resolution bit is set high in the network status register. If half duplex capability is resolved the duplex resolution bit will be low. If the GEM and its link partner cannot resolve a common duplex capability the duplex resolution bit is not set and link will be indicated as being down (bit 2 in the PCS status register and bit 0 in the network status register will both be zero) when auto-negotiation completes. Although the GEM reports the auto-negotiation resolution it does not automatically reconfigure its duplex and pause states. So it is necessary for management software to set the duplex bit in the network configuration register if it reads the duplex resolution bit as being set in the network status register.

PCS Collision Detect and Carrier Sense

The PCS provides both the carrier sense and collision signals for use by the MAC sub-layer when the TBI is active.

CRS (Carrier Sense) is generated by the following conditions:

The receiver has decoded a start of packet/end of packet or receive carrier extension is active. This state is indicated internally to the PCS by CRS receive.

`tx_en` is active, or carrier extension is active for transmit.

The collision signal is generated whenever the PCS is requested to transmit an Ethernet frame at a time coincident with the `crs` receive signal being active. The `col` signal remains active for the duration of the collision. Both `crs` and `col` are asserted, regardless of whether the PCS is operating in half duplex mode or full duplex mode.

Link Status

The PCS link status is indicated on bit 2 of the PCS status register, bit 0 of the network status register and on bit 9 of the interrupt status register. An interrupt is generated each time the PCS link status changes (i.e. link good or link bad).

When auto-negotiation is disabled, the link status value is determined by whether or not the PCS is in a synchronised state. When auto-negotiation is enabled, the link status value is determined by successful completion of auto-negotiation.

Programming Interface

All register addresses are 32 bit word aligned and all read/writes occur over 32-bit data busses. Byte accesses are not supported. Write only (WO) bits return zero if read.

Registers

network_control

Register Information			
Description	"The network control register contains general MAC control functions for both receiver and transmitter."		
Offset	0x000	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	ifg_eats_qav_credit	"Setting this bit high modifies the CBS algorithm so the IFG/IPG associated with a transmit frame counts towards its 802.1Qav credit."	RW	0
29	reserved	reserved	RW	0
28	sel_mii_on_rgmii	"If the RGMII interface being used set this bit high to configure the interface for MII operation."	RW	0
27	oss_correction_field	"1588 One Step Correction Field Update. Set this bit high to enable updating the correction field of PTP 1588 version 2 sync frames by adding current TSU timer value."	RW	0
26	ext_rxq_sel_en	"Enable external selection of receive queue. When this bit is high the ext_match1, ext_match2, ext_match3 and ext_match4 inputs will determine which receive queue a frame is routed to. This will be the case regardless of the state of the external address match enable bit 9 of the network config register. Note that receive frames will be dropped unless they are matched by the internal frame filtering functionality. If the external address match enable bit 9 in the network config register is set frames may be matched by an external address match filter as long as one of the ext_match1, ext_match2, ext_match3 and ext_match4 inputs is asserted early enough. When set ext_rxq_sel_en takes precedence over the existing screener functionality. This bit is only relevant if priority queuing is configured."	RW	0
25	pfc_ctrl	"Enable multiple PFC pause quantums, one per pause priority"	RW	0
24	one_step_sync_mode	"1588 One Step Sync Mode. Write 1 to enable. Replace timestamp field in the 1588 header for TX Sync Frames with current TSU timer value."	RW	0
23	ext_tsu_port_enable	"External TSU timer port enable (1 = enable)"	RW	0
22	store_udp_offset	"Store UDP / TCP offset to memory. Setting this bit to one will cause the upper 16-bits of the CRC of every received frame to be replaced with the offset from start of frame to the beginning of the UDP or TCP header. The lower 16-bits of the CRC are replaced with zero and reserved for future use. The offset is measured in units of 2 bytes. Set to zero for normal operation."	RW	0
21	alt_sgmii_mode	"Alternative sgmii mode. If asserted with sgmii_mode in the network control register the ACK bit is driven before ability detect during transfer of status information from the PHY to the MAC."	RW	0
20	ptp_unicast_ena	"Enable detection of unicast PTP unicast frames."	RW	0

19	tx_lpi_en	"Enable LPI transmission when set LPI (low power idle) is immediately transmitted. Depending on configuration LPI is indicated on the GMII, RGMII or MII interface and can be encoded by the PCS used for SGMII. LPI is transmitted even if bit 3 transmit enable is disabled. Setting this bit also sends a pause signal to the transmit datapath."	RW	0
18	flush_rx_pkt_pclk	"Flush the next packet from the external RX DPRAM. This bit flushes the frame that is the next in line to be pushed out to the AXI interface. Writing one to this bit will only have an effect if the DMA is not currently writing a packet already stored in the DPRAM to memory."	WO	0
17	transmit_pfc_priority_based_pause_frame	"Write a one to transmit PFC priority based pause frame. Takes the values stored in the Transmit PFC Pause Register."	WO	0
16	pfc_enable	"Enable PFC Priority Based Pause Reception capabilities. Setting this bit will enable PFC negotiation and recognition of priority based pause frames. "	RW	0
15	store_rx_ts	"Store receive time stamp to memory. Setting this bit to one will cause the CRC of every received frame to be replaced with the value of the nanoseconds field of the 1588 timer that was captured as the receive frame passed the message time stamp point. Set to zero for normal operation."	RW	0
14	stats_read_snap	"Read snapshot - writing a one means that the snapshot value of the statistics register will be read back, otherwise the raw statistic register will be read. The default GEM configuration does not support this function. See Parameterization section under Implementation Application Notes for more details."	RW	0
13	stats_take_snap	"Take snapshot - writing a one will record the current value of all statistics registers in the snapshot registers and clear the statistics registers. The default GEM configuration does not support this function."	WO	0
12	tx_pause_frame_zero	"Transmit zero quantum pause frame - writing one to this bit causes a pause frame with zero quantum to be transmitted."	WO	0
11	tx_pause_frame_req	"Transmit pause frame - writing one to this bit causes a pause frame to be transmitted."	WO	0
10	tx_halt_pclk	"Transmit halt - writing one to this bit halts transmission as soon as any ongoing frame transmission ends."	WO	0
9	tx_start_pclk	"Start transmission - writing one to this bit starts transmission."	WO	0
8	back_pressure	"Back pressure if set in 10M or 100M half duplex mode will force collisions on all received frames. Ignored in gigabit half duplex mode."	RW	0
7	stats_write_en	"Write enable for statistics registers - setting this bit to one means the statistics registers can be written for functional test purposes."	RW	0
6	inc_all_stats_regs	"Incremental statistics registers - this bit is write only. Writing a one increments all the statistics registers by one for test purposes."	WO	0
5	clear_all_stats_regs	"Clear statistics registers - this bit is write only. Writing a one clears the statistics registers."	WO	0

4	man_port_en	"Management port enable - set to one to enable the management port. When zero forces mdio to high impedance state and mdc low."	RW	0
3	enable_transmit	"Transmit enable - when set, it enables the GEM transmitter to send data. When reset transmission will stop immediately, the transmit pipeline and control registers will be cleared and the transmit queue pointer register will reset to point to the start of the transmit descriptor list."	RW	0
2	enable_receive	"Receive enable - when set, it enables the GEM to receive data. When reset frame reception will stop immediately and the receive pipeline will be cleared. The receive queue pointer register is unaffected."	RW	0
1	loopback_local	"Loopback local - asserts the loopback_local signal to the system clock generator. Also connects txd to rxd, tx_en to rx_dv and forces full duplex mode. Bit 11 of the network configuration register must be set low to disable TBI mode when in internal loopback. rx_clk and tx_clk may malfunction as the GEM is switched into and out of internal loopback. It is important that receive and transmit circuits have already been disabled when making the switch into and out of internal loopback. Local loopback functionality is optional and may not be supported by some instantiations of the GEM."	RW	0
0	loopback	"Loopback - controls the loopback output pin."	RW	0

network_config

Register Information			
Description	"The network configuration register contains functions for setting the mode of operation for the Gigabit Ethernet MAC."		
Offset	0x004	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31	uni_direction_enable	"Uni-direction-enable. When low the PCS will transmit idle symbols if the link goes down. When high the PCS can transmit frame data when the link is down."	RW	0
30	ignore_ipg_rx_er	"Ignore IPG rx_er. When set rx_er has no effect on the GEMs operation when rx_dv is low. Set this when using the RGMII wrapper in half-duplex mode."	RW	0
29	nsp_change	"Receive bad preamble. When set frames with non-standard preamble are not rejected."	RW	0
28	ipg_stretch_enable	"IPG stretch enable - when set the transmit IPG can be increased above 96 bit times depending on the previous frame length using the IPG stretch register."	RW	0
27	sgmii_mode_enable	"SGMII mode enable - changes behaviour of the auto-negotiation advertisement and link partner ability registers to meet the requirements of SGMII and reduces the duration of the link timer from 10 ms to 1.6 ms."	RW	0
26	ignore_rx_fcs	"Ignore RX FCS - when set frames with FCS/CRC errors will not be rejected. FCS error statistics will still be collected for frames with bad FCS and FCS status will be recorded in frame's DMA descriptor. For normal operation this bit must be set to zero."	RW	0
25	en_half_duplex_rx	"Enable frames to be received in half-duplex mode while transmitting."	RW	0
24	receive_checksum_offload_enable	"Receive checksum offload enable - when set, the receive checksum engine is enabled. Frames with bad IP, TCP or UDP checksums are discarded."	RW	0
23	disable_copy_of_pause_frames	"Disable copy of pause frames - set to one to prevent pause frames being copied to memory. When set, neither control frames with type id 8808, nor pause frames with destination address 010000c28001 are copied to memory, this functionality was enhanced in release 1p09. Note that valid pause frames received will still increment pause statistics and pause the transmission of frames as required."	RW	0
22:21	data_bus_width	"Data bus width - set according to AMBA (AHB/AXI) or external FIFO data bus width. The reset value for this can be changed by defining a new value for gem_dma_bus_width_def in gem_defs. Only valid bus widths may be written if the system is configured to a maximum width less than 128-bits. 00: 32 bit data bus width 01: 64 bit AMBA (AHB/AXI) data bus width 10: 128 bit AMBA (AHB/AXI) data bus width 11: 128 bit AMBA (AHB/AXI) data bus width. Note. The reset value of this field is equal to the gem_dma_bus_width_def define, which is user configurable."	RW	0x0

20:18	mdc_clock_division	"MDC clock division - set according to pclk speed. These three bits determine the number pclk will be divided by to generate MDC. For conformance with the 802.3 specification, MDC must not exceed 2.5 MHz (MDC is only active during MDIO read and write operations). The reset value for this can be changed by defining a new value for gem_mdc_clock_div in gem_defs.v 000: divide pclk by 8 (pclk up to 20 MHz) 001: divide pclk by 16 (pclk up to 40 MHz) 010: divide pclk by 32 (pclk up to 80 MHz) 011: divide pclk by 48 (pclk up to 120MHz) 100: divide pclk by 64 (pclk up to 160 MHz) 101: divide pclk by 96 (pclk up to 240 MHz) 110: divide pclk by 128 (pclk up to 320 MHz) 111: divide pclk by 224 (pclk up to 540 MHz). Note. The reset value of this field is equal to the gem_mdc_clock_div define, which is user configurable."	RW	0x2
17	fcs_remove	"FCS remove - setting this bit will cause received frames to be written to memory without their frame check sequence (last 4 bytes). The frame length indicated will be reduced by four bytes in this mode."	RW	0
16	length_field_error_frame_discard	"Length field error frame discard - setting this bit causes frames with a measured length shorter than the extracted length field (as indicated by bytes 13 and 14 in a non-VLAN tagged frame) to be discarded. This only applies to frames with a length field less than 0x0600."	RW	0
15:14	receive_buffer_offset	"Receive buffer offset - indicates the number of bytes by which the received data is offset from the start of the receive buffer. Note that when the define gem_pbuf_rsc has been set then these bits cannot be used."	RW	0x0
13	pause_enable	"Pause enable - when set, transmission will pause if a non zero 802.3 classic pause frame is received and PFC has not been negotiated."	RW	0
12	retry_test	"Retry test - must be set to zero for normal operation. If set to one the backoff between collisions will always be one slot time. Setting this bit to one helps test the too many retries condition. Also used in the pause frame tests to reduce the pause counter's decrement time from 512 bit times, to every rx_clk cycle."	RW	0
11	pcs_select	"PCS select - selects between MII/GMII and TBI. Must be set for SGMII operation. 0: GMII/MII interface enabled, TBI disabled 1: TBI enabled, GMII/MII disabled"	RW	0
10	gigabit_mode_enable	"Gigabit mode enable - setting this bit configures the GEM for 1000 Mbps operation. 0: 10/100 operation using MII or TBI interface 1: Gigabit operation using GMII or TBI interface"	RW	0
9	external_address_match_enable	"External address match enable - when set the external address match interface can be used to copy frames to memory."	RW	0
8	receive_1536_byte_frames	"Receive 1536 byte frames - setting this bit means the GEM will accept frames up to 1536 bytes in length. Normally the GEM would reject any frame above 1518 bytes."	RW	0
7	unicast_hash_enable	"Unicast hash enable - when set, unicast frames will be accepted when the 6 bit hash function of the destination address points to a bit that is set in the hash register."	RW	0

6	multicast_hash_enable	"Multicast hash enable - when set, multicast frames will be accepted when the 6 bit hash function of the destination address points to a bit that is set in the hash register."	RW	0
5	no_broadcast	"No broadcast - when set to logic one, frames addressed to the broadcast address of all ones will not be accepted."	RW	0
4	copy_all_frames	"Copy all frames - when set to logic one, all valid frames will be accepted."	RW	0
3	jumbo_frames	"Jumbo frames - set to one to enable jumbo frames up to `gem_jumbo_max_length bytes to be accepted. The default length is 10,240 bytes."	RW	0
2	discard_non_vlan_frames	"Discard non-VLAN frames - when set only VLAN tagged frames will be passed to the address matching logic."	RW	0
1	full_duplex	"Full duplex - if set to logic one, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting. Also controls the half_duplex pin."	RW	0
0	speed	"Speed - set to logic one to indicate 100Mbps operation, logic zero for 10Mbps. The value of this pin is reflected on the speed_mode[0] output pin."	RW	0

network_status

Register Information			
Description	"The network status register returns status information with respect to the PHY management interface."		
Offset	0x008	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7	lpi_indicate_pclk	"LPI Indication - Low power idle has been detected on receive. This bit is set when LPI is detected and reset when normal idle is detected. An interrupt is generated when the state of this bit changes."	RO	0
6	pfc_negotiate_pclk	"Set when PFC Priority Based Pause has been negotiated."	RO	0
5	mac_pause_tx_en	"PCS auto-negotiation pause transmit resolution."	RO	0
4	mac_pause_rx_en	"PCS auto-negotiation pause receive resolution"	RO	0
3	mac_full_duplex	"PCS auto-negotiation duplex resolution. Set to one if the resolution function determines that both devices are capable of full duplex operation. If zero half-duplex operation is possible as long as bit 0 (PCS link state) is also one."	RO	0
2	man_done	"The PHY management logic is idle (i.e. has completed)."	RO	1
1	mdio_in	"Returns status of the mdio_in pin."	RO	0
0	pcs_link_state	"Returns status of PCS link state. If auto-negotiation is disabled this returns the synchronisation status. If auto-negotiation is enabled it is set in the LINK_OK state as long as a compatible duplex mode is resolved, it is always set in the LINK_OK state in SGMII mode."	RO	0

user_io_register

Register Information			
Description	"The GEM design provides up to 16 inputs and 16 outputs so that the I/O may be read or set under the control of the processor interface. If the user I/O is disabled as a configuration option, this address space is defined as reserved, and hence will be a read-only register of the value 0x0. If enabled, the number of inputs and outputs can be configured separately. The first output will be represented in bit 0 of the user I/O register, the second output will use bit 1 and so on. The first input will be represented in bit 16 of the user I/O register, the second input will use bit 17 and so on."		
Offset	0x00C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	user_programmable_inputs	"User programmable inputs - the upper 16 bits of this register are used to monitor the state of the user inputs. A logic one read from a bit in this range will correspond to the input being in a high state. A logic zero read from a bit in this range will correspond to the input being in a low state. Any unused bits will be read as 0. Writing to any bits in this range will have no functional effect."	RO	0x0000
15:0	user_programmable_outputs	"User programmable outputs - the lower 16 bits of this register are used to control the state of the user outputs. A logic one written to a bit in this range will cause the corresponding output to be set high. A logic zero written to a bit in this range shall cause the corresponding output to be forced low. Any unused bits will be read as logic zero. Writing to any unused bits in this range will have no functional effect."	RW	0x0000

dma_config

Register Information			
Description	"DMA Configuration Register"		
Offset	0x010	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	dma_addr_bus_width_1	"DMA address bus width. 0 = 32b, 1 = 64b."	RW	0
29	tx_bd_extended_mode_en	"Enable TX extended BD mode. See TX BD control register definition for description of feature."	RW	0
28	rx_bd_extended_mode_en	"Enable RX extended BD mode. See RX BD control register definition for description of feature."	RW	0
27	reserved_27	"Reserved, read as 0, ignored on write."	RO	0
26	force_max_amba_burst_tx	"Force max length bursts on TX. Force the TX DMA to always issue max length bursts on EOP(end of packet) or EOB(end of buffer) transfers as defined by bits 4:0 of this register, even when there is less that max burst data bytes to read. Residual data read is ignored. AHB only - does not apply on AXI bursts or bursts that break 1k boundary rule."	RW	0
25	force_max_amba_burst_rx	"Force max length bursts on RX. Force the RX DMA to always issue max length bursts on EOP(end of packet) or EOB(end of buffer)transfers, even if there is less than max burst real packet data required to write. Any extra bytes of pad data is set to 0x00. AHB only - does not apply on AXI bursts or bursts that break 1k boundary rule."	RW	0
24	force_discard_on_err	"Auto Discard RX pkts during lack of resource. When set, the GEM DMA will automatically discard receive packets from the receiver packet buffer memory when no AMBA (AHB/AXI) resource is available. When low, then received packets will remain to be stored in the SRAM based packet buffer until AMBA (AHB/AXI) buffer resource next becomes available. A write to this bit is ignored if the DMA is not configured in the packet buffer full store and forward mode."	RW	0
23:16	rx_buf_size	"DMA receive buffer size in external AMBA (AHB/AXI) system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte. 0x18: 1536 byte (1*max length frame/buffer)0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02
15:14	reserved_15_14	"Reserved, read as 0, ignored on write."	RO	0x0

13	crc_error_report	"When the bit is set, bit 16 of the receive buffer descriptor will represent FCS/CRC error (only if frames with FCS are copied to memory as enabled by bit 26 in the network config register). When this bit is clear, bit 16 of the receive buffer descriptor will represent the Canonical format indicator (CFI) bit as extracted from the receive frame (if the receive buffer descriptor is pointing to the last data buffer of the receive frame and the received frame was VLAN tagged)."	RW	0
12	infinite_last_dbuf_size_en	"Forces the receive DMA to consider the data buffer pointed to by last descriptor in the descriptor list to be of infinite size."	RW	0
11	tx_pbuf_tcp_en	"Transmitter IP, TCP and UDP checksum generation offload enable (not supported when in TX Partial Store and Forward mode). When set, the transmitter checksum generation engine is enabled, to calculate and substitute checksums for transmit frames. When clear, frame data is unaffected. If the GEM is not configured to use the DMA packet buffer, this bit is not implemented and will be treated as reserved, read as 0, ignored on write."	RW	0
10	tx_pbuf_size	"Transmitter packet buffer memory size select. Having this bit at zero halves the amount of memory used for the transmit packet buffer. This reduces the amount of memory used by the GEM. It is important to set this bit to one if the full configured physical memory is available. The value in brackets below represents the size that would result for the default maximum configured memory size of 4 Kbytes. 1: Use full configured addressable space (4 Kb) 0: Do not use top address bit (2 Kb) If the GEM is not configured to use the DMA packet buffer, this bit is not implemented and will be treated as reserved, read as 0, ignored on write. Note. The reset value of this field is equal to the gem_tx_pbuf_size_def define, which is user configurable."	RW	1
9:8	rx_pbuf_size	"Receiver packet buffer memory size select. Having these bits at less than 11 reduces the amount of memory used for the receive packet buffer. This reduces the amount of memory used by the GEM. It is important to set these bits both to one if the full configured physical memory is available. The value in brackets below represents the size that would result for the default maximum configured memory size of 8 Kbytes. 11: Use full configured addressable space (8 Kb) 10: Do not use top address bit (4 Kb) 01: Do not use top two address bits (2 Kb) 00: Do not use top three address bits (1 Kb) If the GEM is not configured to use the DMA packet buffer, these bits are not implemented and will be treated as reserved, read as 0, ignored on write. Note. The reset value of this field is equal to the gem_rx_pbuf_size_def define, which is user configurable."	RW	0x3
7	endian_swap_packet	"endian swap mode enable for packet data accesses. When set, selects swapped endianism for AMBA (AHB/AXI) transfers. When clear, selects little endian mode."	RW	1

6	endian_swap_management	"endian swap mode enable for management descriptor accesses. When set, selects swapped endianism for AMBA (AHB/AXI) transfers. When clear, selects little endian mode."	RW	1
5	hdr_data_splitting_en	"Enable header data Splitting. When set, receive frames will be forwarded to main memory using a minimum of two DMA data buffers. The first X data buffers will contain the frame header, consisting of the Ethernet,VLAN,(IPv4 or IPv6),(TCP or UDP). X= (frame header size divided by rx_buf_size as defined in bits 23:16 of this register). The last Y data buffers will contain the frame payload. Y= (frame payload size divided by rx_buf_size). Note that for non VLAN/IP/TCP/UDP frames, the header will always be 14 bytes. When this feature is disabled, the frame is forwarded to main memory in blocks of rx_buf_size."	RW	0
4:0	amba_burst_length	"Selects the burst length to use on the AMBA (AHB/AXI) when transferring frame data. Not used for DMA management operations and only used where space and data size allow and respecting AXI/AHB burst boundary rules. One-hot priority encoding enforced automatically on register writes as follows, where x represents don't care: 1xxxx: Attempt to use bursts of up to 16. 01xxx: Attempt to use bursts of up to 8. 001xx: Attempt to use bursts of up to 4. 0001x: Always use SINGLE bursts. 00001: Always use SINGLE bursts. 00000: Attempt to use bursts of up to 256."	RW	0x04

transmit_status

Register Information			
Description	"This register, when read, provides details of the status of a transmit. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register."		
Offset	0x014	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:9	reserved_31_9	"Reserved, read as 0, ignored on write."	RO	0x00 0000
8	resp_not_ok	"bresp/hresp not OK - set when the DMA block sees bresp/hresp not OK. Cleared by writing a one to this bit."	RW W1toClr	0
7	late_collision_occurred	"Late collision occurred - only set if the condition occurs in gigabit mode, as retry is not attempted. Cleared by writing a one to this bit."	RW W1toClr	0
6	transmit_under_run	"Transmit under run - this bit is set if the transmitter was forced to terminate a frame that it had already began transmitting due to further data being unavailable. This bit is set if a transmitter status write back has not completed when another status write back is attempted. When using the DMA interface configured for internal FIFO mode, this bit is also set when the transmit DMA has written the SOP data into the FIFO and either the AHB bus was not granted in time for further data, or because an AHB not OK response was returned, or because a used bit was read. When using the DMA interface configured for packet buffer mode, this bit will never be set. When using the external FIFO interface, this bit is also set when the tx_r_underflow input is asserted during a frame transfer. Cleared by writing a 1."	RW W1toClr	0
5	transmit_complete	"Transmit complete - set when a frame has been transmitted. Cleared by writing a one to this bit."	RW W1toClr	0
4	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) errors. Set if an error occurs whilst midway through reading transmit frame from external memory including HRESP(AHB), RRESP or BRESP(AXI) errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and tx_er asserted). Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size. Cleared by writing a one to this bit."	RW W1toClr	0
3	transmit_go	"Transmit go - if high transmit is active. When using the exposed FIFO interface, this bit represents bit 3 of the network control register. When using the DMA interface this bit represents the tx_go variable as specified in the transmit buffer description."	RO	0
2	retry_limit_exceeded	"Retry limit exceeded - cleared by writing a one to this bit."	RW W1toClr	0
1	collision_occurred	"Collision occurred - set by the assertion of collision. Cleared by writing a one to this bit. When operating in 10/100 mode, this status indicates either a collision or a late collision. In gigabit mode, this status is not set for a late collision."	RW W1toClr	0
0	used_bit_read	"Used bit read - set when a transmit buffer descriptor is read with its used bit set. Cleared by writing a one to this bit."	RW W1toClr	0

receive_q_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x018	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

transmit_q_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x01C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

receive_status

Register Information			
Description	"This register, when read provides details of the status of a receive. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register."		
Offset	0x020	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:4	reserved_31_4	"Reserved, read as 0, ignored on write."	RO	0x000 0000
3	resp_not_ok	"bresp/hresp not OK - set when the DMA block sees bresp/hresp not OK. Cleared by writing a one to this bit."	RW W1toClr	0
2	receive_overrun	"Receive overrun - this bit is set if either the gem_dma RX FIFO or external RX FIFO were unable to store the receive frame due to a FIFO overflow, or if the receive status, reported by the gem_rx module to the gem_dma was not taken at end of frame. This bit is also set in DMA packet buffer mode if the packet buffer overflows. For DMA operation the buffer will be recovered if an overrun occurs. This bit is cleared by writing a one to it."	RW W1toClr	0
1	frame_received	"Frame received - one or more frames have been received and placed in memory. Cleared by writing a one to this bit."	RW W1toClr	0
0	buffer_not_available	"Buffer not available - an attempt was made to get a new buffer and the pointer indicated that it was owned by the processor. The DMA will reread the pointer each time an end of frame is received until a valid pointer is found. This bit is set following each descriptor read attempt that fails, even if consecutive pointers are unsuccessful and software has in the mean time cleared the status flag. Cleared by writing a one to this bit."	RW W1toClr	0

int_status

Register Information			
Description	"If not configured for priority queueing, the GEM generates a single interrupt. This register indicates the source of this interrupt. The corresponding bit in the mask register must be clear for a bit to be set. If any bit is set in this register the ethernet_int signal will be asserted. For test purposes each bit can be set or reset by writing to the interrupt mask register. The default configuration is shown below whereby all bits are reset to zero on read. Changing the validity of the `gem_irq_read_clear define will instead require a one to be written to the appropriate bit in order to clear it. In this mode reading has no affect on the status of the bit."		
Offset	0x024	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	reserved_31_31	"Reserved, read as 0, ignored on write."	RO	0
30	reserved_30_30	"Reserved, read as 0, ignored on write."	RO	0
29	tsu_timer_comparison_interrupt	"TSU timer comparison interrupt. Indicates when TSU timer count value is equal to programmed value."	RW RtoClr	0
28	wol_interrupt	"WOL interrupt. Indicates a WOL event has been received."	RW RtoClr	0
27	receive_lpi_indication_status_bit_change	"Receive LPI indication status bit change"	RW RtoClr	0
26	tsu_seconds_register_increment	"TSU seconds register increment indicates the register has incremented. Cleared on read."	RW RtoClr	0
25	ptp_pdelay_resp_frame_transmitted	"PTP pdelay_resp frame transmitted indicates a PTP pdelay_resp frame has been transmitted. Cleared on read."	RW RtoClr	0
24	ptp_pdelay_req_frame_transmitted	"PTP pdelay_req frame transmitted indicates a PTP pdelay_req frame has been transmitted. Cleared on read."	RW RtoClr	0
23	ptp_pdelay_resp_frame_received	"PTP pdelay_resp frame received indicates a PTP pdelay_resp frame has been received. Cleared on read."	RW RtoClr	0
22	ptp_pdelay_req_frame_received	"PTP pdelay_req frame received indicates a PTP pdelay_req frame has been received. Cleared on read."	RW RtoClr	0
21	ptp_sync_frame_transmitted	"PTP sync frame transmitted indicates a PTP sync frame has been transmitted. Cleared on read."	RW RtoClr	0
20	ptp_delay_req_frame_transmitted	"PTP delay_req frame transmitted indicates a PTP delay_req frame has been transmitted. Cleared on read."	RW RtoClr	0
19	ptp_sync_frame_received	"PTP sync frame received indicates a PTP sync frame has been received. Cleared on read."	RW RtoClr	0
18	ptp_delay_req_frame_received	"PTP delay_req frame received indicates a PTP delay_req frame has been received. Cleared on read."	RW RtoClr	0

17	pcs_link_partner_page_received	"PCS link partner page received - set when a new base page or next page is received from the link partner. The first time this interrupt is received, it will indicate base page received and subsequent reads will indicate next pages. The next page and base page registers should only be read when this interrupt is signalled. For next pages, the link partner next page register should be read first to avoid the register being over written. This interrupt also indicates when the host should write a new page into the next page register. If further next page exchange is only required by the link partner, this register should be written with a null message page (0x2001). Cleared on read."	RW RtoClr	0
16	pcs_auto_negotiation_complete	"PCS auto-negotiation complete - set once the internal PCS layer has completed auto-negotiation. Cleared on read."	RW RtoClr	0
15	external_interrupt	"External interrupt - set when a rising edge has been detected on the ext_interrupt_in input pin. Cleared on read."	RW RtoClr	0
14	pause_frame_transmitted	"Pause frame transmitted - indicates a pause frame has been successfully transmitted after being initiated from the network control register or from the tx_pause control pin. Cleared on read."	RW RtoClr	0
13	pause_time_elapsed	"Pause Time elapsed - set when either the pause time register at address 0x38 decrements to zero, or when a valid pause frame is received with a zero pause quantum field. Not set for PFC pause frames. Cleared on read."	RW RtoClr	0
12	pause_frame_with_non_zero_pause_quantum_received	"Pause frame with non-zero pause quantum received - indicates a valid pause has been received that has a non-zero pause quantum field. Cleared on read."	RW RtoClr	0
11	resp_not_ok	"bresp/hresp not OK - set when the DMA block sees bresp/hresp not OK. Cleared on read."	RW RtoClr	0
10	receive_overrun	"Receive overrun - set when the receive overrun status bit gets set. Cleared on read."	RW RtoClr	0
9	link_change	"Link change - set when the state of the link detected by the internal PCS changes state. Cleared on read."	RW RtoClr	0
8	reserved_8	"Reserved, read as 0, ignored on write."	RO	0
7	transmit_complete	"Transmit complete - set when a frame has been transmitted. Cleared on read."	RW RtoClr	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error. Set if an error occurs whilst midway through reading transmit frame from external system memory, including HRESP errors(AHB), RRESP or BRESP(AXI) errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and tx_er asserted). Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size. Cleared on a read."	RW RtoClr	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision - transmit error. Late collision will only cause this status bit to be set in gigabit mode (as a retry is not attempted). Cleared on read."	RW RtoClr	0

4	transmit_under_run	"Transmit under run - this interrupt is set if the transmitter was forced to terminate a frame that it has already began transmitting due to further data being unavailable. If an under run occurs, the transmitter will force bad crc and tx_er high. This interrupt is set if a transmitter status write back has not completed when another status write back is attempted. When using the DMA interface configured for internal FIFO mode, this interrupt is also set when the transmit DMA has written the SOP data into the FIFO and either the AHB bus was not granted in time for further data, or because an AHB/AXI error response was returned by the connected slave, or because a used bit was read. When using the DMA interface configured for packet buffer mode, this bit will never be set. When using the external FIFO interface, this interrupt is also set when the tx_r_underflow input was asserted during a frame transfer. Cleared on read."	RW RtoClr	0
3	tx_used_bit_read	"TX used bit read - set when a transmit buffer descriptor is read with its used bit set. Cleared on read."	RW RtoClr	0
2	rx_used_bit_read	"RX used bit read - set when a receive buffer descriptor is read with its used bit set. Cleared on read."	RW RtoClr	0
1	receive_complete	"Receive complete - a frame has been stored in memory. Cleared on read."	RW RtoClr	0
0	management_frame_sent	"Management frame sent - the PHY maintenance register has completed its operation. Cleared on read."	RW RtoClr	0

int_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x028	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31_31	"Reserved, read as 0, ignored on write."	RO	0
30	reserved_30_30	"Reserved, read as 0, ignored on write."	RO	0
29	enable_tsu_timer_comparison_interrupt	"Enable TSU timer comparison interrupt."	WO	0
28	enable_wol_event_received_interrupt	"Enable WOL event received interrupt"	WO	0
27	enable_rx_lpi_indication_interrupt	"Enable RX LPI indication interrupt"	WO	0
26	enable_tsu_seconds_register_increment	"Enable TSU seconds register increment"	WO	0
25	enable_ptp_pdelay_resp_frame_transmitted	"Enable PTP pdelay_resp frame transmitted"	WO	0
24	enable_ptp_pdelay_req_frame_transmitted	"Enable PTP pdelay_req frame transmitted"	WO	0
23	enable_ptp_pdelay_resp_frame_received	"Enable PTP pdelay_resp frame received"	WO	0
22	enable_ptp_pdelay_req_frame_received	"Enable PTP pdelay_req frame received"	WO	0
21	enable_ptp_sync_frame_transmitted	"Enable PTP sync frame transmitted "	WO	0
20	enable_ptp_delay_req_frame_transmitted	"Enable PTP delay_req frame transmitted "	WO	0
19	enable_ptp_sync_frame_received	"Enable PTP sync frame received"	WO	0
18	enable_ptp_delay_req_frame_received	"Enable PTP delay_req frame received"	WO	0
17	enable_pcs_link_partner_page_received	"Enable PCS link partner page received"	WO	0
16	enable_pcs_auto_negotiation_complete_interrupt	"Enable PCS auto-negotiation complete interrupt"	WO	0
15	enable_external_interrupt	"Enable external interrupt"	WO	0
14	enable_pause_frame_transmitted_interrupt	"Enable pause frame transmitted interrupt"	WO	0

13	enable_pause_time_zero_interrupt	"Enable pause time zero interrupt"	WO	0
12	enable_pause_frame_with_non_zero_pause_quantum_interrupt	"Enable pause frame with non-zero pause quantum interrupt"	WO	0
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10	enable_receive_overrun_interrupt	"Enable receive overrun interrupt"	WO	0
9	enable_link_change_interrupt	"Enable link change interrupt"	WO	0
8	not_used	"Not used"	RO	0
7	enable_transmit_complete_interrupt	"Enable transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable retry limit exceeded or late collision interrupt"	WO	0
4	enable_transmit_buffer_under_run_interrupt	"Enable transmit buffer under run interrupt"	WO	0
3	enable_transmit_used_bit_read_interrupt	"Enable transmit used bit read interrupt"	WO	0
2	enable_receive_used_bit_read_interrupt	"Enable receive used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable receive complete interrupt"	WO	0
0	enable_management_done_interrupt	"Enable management done interrupt"	WO	0

int_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x02C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31_31	"Reserved, read as 0, ignored on write."	RO	0
30	reserved_30_30	"Reserved, read as 0, ignored on write."	RO	0
29	disable_tsu_timer_comparison_interrupt	"Disable TSU timer comparison interrupt."	WO	0
28	disable_wol_event_received_interrupt	"Disable WOL event received interrupt"	WO	0
27	disable_rx_lpi_indication_interrupt	"Disable RX LPI indication interrupt"	WO	0
26	disable_tsu_seconds_register_increment	"Disable TSU seconds register increment"	WO	0
25	disable_ptp_pdelay_resp_frame_transmitted	"Disable PTP pdelay_resp frame transmitted"	WO	0
24	disable_ptp_pdelay_req_frame_transmitted	"Disable PTP pdelay_req frame transmitted"	WO	0
23	disable_ptp_pdelay_resp_frame_received	"Disable PTP pdelay_resp frame received"	WO	0
22	disable_ptp_pdelay_req_frame_received	"Disable PTP pdelay_req frame received"	WO	0
21	disable_ptp_sync_frame_transmitted	"Disable PTP sync frame transmitted "	WO	0
20	disable_ptp_delay_req_frame_transmitted	"Disable PTP delay_req frame transmitted "	WO	0
19	disable_ptp_sync_frame_received	"Disable PTP sync frame received"	WO	0
18	disable_ptp_delay_req_frame_received	"Disable PTP delay_req frame received"	WO	0
17	disable_pcs_link_partner_page_received	"Disable PCS link partner page received"	WO	0
16	disable_pcs_auto_negotiation_complete_interrupt	"Disable PCS auto-negotiation complete interrupt"	WO	0
15	disable_external_interrupt	"Disable external interrupt"	WO	0
14	disable_pause_frame_transmitted_interrupt	"Disable pause frame transmitted interrupt"	WO	0

13	disable_pause_time_zero_interrupt	"Disable pause time zero interrupt"	WO	0
12	disable_pause_frame_with_non_zero_pause_quantum_interrupt	"Disable pause frame with non-zero pause quantum interrupt"	WO	0
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10	disable_receive_overrun_interrupt	"Disable receive overrun interrupt"	WO	0
9	disable_link_change_interrupt	"Disable link change interrupt"	WO	0
8	not_used	"Not used"	RO	0
7	disable_transmit_complete_interrupt	"Disable transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable retry limit exceeded or late collision interrupt"	WO	0
4	disable_transmit_buffer_under_run_interrupt	"Disable transmit buffer under run interrupt"	WO	0
3	disable_transmit_used_bit_read_interrupt	"Disable transmit used bit read interrupt"	WO	0
2	disable_receive_used_bit_read_interrupt	"Disable receive used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable receive complete interrupt"	WO	0
0	disable_management_done_interrupt	"Disable management done interrupt"	WO	0

int_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x030	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31_31	"Reserved, read as 0, ignored on write."	RO	0
30	reserved_30_30	"Reserved, read as 0, ignored on write."	RO	0
29	tsu_timer_comparison_mask	"Enable TSU timer comparison interrupt mask."	RO	1
28	wol_event_received_mask	"A read of this register returns the value of the WOL event received mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
27	rx_lpi_indication_mask	"A read of this register returns the value of the RX LPI indication mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written"	RO	1
26	tsu_seconds_register_increment_mask	"A read of this register returns the value of the TSU seconds register increment mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
25	ptp_pdelay_resp_frame_transmitted_mask	"A read of this register returns the value of the PTP pdelay_resp frame transmitted mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
24	ptp_pdelay_req_frame_transmitted_mask	"A read of this register returns the value of the PTP pdelay_req frame transmitted mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
23	ptp_pdelay_resp_frame_received_mask	"A read of this register returns the value of the PTP pdelay_resp frame received mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1

22	ptp_pdelay_req_frame_received_mask	"A read of this register returns the value of the PTP pdelay_req frame received mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
21	ptp_sync_frame_transmitted_mask	"A read of this register returns the value of the PTP sync frame transmitted mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
20	ptp_delay_req_frame_transmitted_mask	"A read of this register returns the value of the PTP delay_req frame transmitted mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
19	ptp_sync_frame_received_mask	"A read of this register returns the value of the PTP sync frame received mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
18	ptp_delay_req_frame_received_mask	"A read of this register returns the value of the PTP delay_req frame received mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
17	pcs_link_partner_page_mask	"A read of this register returns the value of the PCS link partner page mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
16	pcs_auto_negotiation_complete_interrupt_mask	"A read of this register returns the value of the PCS auto-negotiation complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
15	external_interrupt_mask	"external interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
14	pause_frame_transmitted_interrupt_mask	"pause frame transmitted interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1

13	pause_time_zero_interrupt_mask	"pause time zero interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
12	pause_frame_with_non_zero_pause_quantum_interrupt_mask	"pause frame with non-zero pause quantum interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10	receive_overrun_interrupt_mask	"receive overrun interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
9	link_change_interrupt_mask	"A read of this register returns the value of the link change interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
8	not_used	"Not used"	RO	1
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"transmit frame corruption due to AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_mask	"A read of this register returns the value of the retry limit exceeded or late collision (gigabit mode only) interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4	transmit_buffer_under_run_interrupt_mask	"transmit buffer under run interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1

3	transmit_used_bit_read_interrupt_mask	"transmit used bit read interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
2	receive_used_bit_read_interrupt_mask	"receive used bit read interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	management_done_interrupt_mask	"management done interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1

phy_management

Register Information			
Description	"The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit-2 is set in the network status register. It takes about 2000 pclk cycles to complete, when MDC is set for pclk divide by 32 in the network configuration register. An interrupt is generated upon completion. During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each MDC cycle. This causes transmission of a PHY management frame on MDIO. See Section 22.2.4.5 of the IEEE 802.3 standard. Reading during the shift operation will return the current contents of the shift register. At the end of management operation, the bits will have shifted back to their original locations. For a read operation, the data bits will be updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced. The MDIO interface can read IEEE 802.3 clause 45 PHYs as well as clause 22 PHYs. To read clause 45 PHYs, bit 30 should be written with a 0 rather than a 1. For a description of MDC generation, see Network Configuration Register."		
Offset	0x034	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	write0	"Must be written with 0."	RW	0
30	write1	"Must be written to 1 for a valid Clause 22 frame and to 0 for a valid Clause 45 frame."	RW	0
29:28	operation	"Operation. For a Clause 45 frame: 00 is an addr, 01 is a write, 10 is a post read increment, 11 is a read frame. For a Clause 22 frame: 10 is a read, 01 is a write."	RW	0x0
27:23	phy_address	"PHY address."	RW	0x00
22:18	register_address	"Register address - specifies the register in the PHY to access."	RW	0x00
17:16	write10	"Must be written with 10."	RW	0x0
15:0	phy_write_read_data	"For a write operation this is written with the data to be written to the PHY. After a read operation this contains the data read from the PHY."	RW	0x0000

pause_time

Register Information			
Description	"Received Pause Quantum Register"		
Offset	0x038	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	quantum	"Received pause quantum - stores the current value of the received pause quantum register which is decremented every 512 bit times."	RO	0x0000

tx_pause_quantum

Register Information				
Description	"Transmit Pause Quantum Register"			
Offset	0x03C		**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	quantum_p1	"Transmit pause quantum - written with the pause quantum value for pause frame transmission of priority 1."	RW	0xFFFF
15:0	quantum	"Transmit pause quantum - written with the pause quantum value for pause frame transmission."	RW	0xFFFF

pbuf_txcutthru

Register Information				
Description	"Partial store and forward is only applicable when using the DMA configured in SRAM based packet buffer mode. It is also not available when using multi buffer frames. TX Partial Store and Forward"			
Offset	0x040		**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	dma_tx_cutthru	"Enable TX partial store and forward operation"	RW	0
30:14	reserved	"Reserved, read as 0, ignored on write."	RO	0x0 0000
13:0	dma_tx_cutthru_threshold	"Watermark value. This value must be >= 0x14. The reset value depends on the value of the configuration option `gem_tx_pbuf_addr, which is defined in the verilog defs configuration file. The value chosen for the generation of the userguide was `gem_tx_pbuf_addr = 14"	RW	0x3FFF

pbuf_rxcutthru

Register Information				
Description	"RX Partial Store and Forward"			
Offset	0x044		**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	dma_rx_cutthru	"Enable RX partial store and forward operation"	RW	0
30:11	reserved	"Reserved, read as 0, ignored on write."	RO	0x0 0000
10:0	dma_rx_cutthru_threshold	"Watermark value. The reset value depends on the value of the configuration option `gem_rx_pbuf_addr, which is defined in the verilog defs configuration file. The value chosen for the generation of the userguide was `gem_rx_pbuf_addr = 11."	RW	0x7FF

jumbo_max_length

Register Information					
Description	"Maximum Jumbo Frame Size."				
Offset	0x048		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:14	reserved_31_14	"Reserved, read as 0, ignored on write."		RO	0x0 0000
13:0	jumbo_max_length	"Maximum Jumbo Frame Size - resets to the gem_jumbo_max_length define value."		RW	0x2800

external_fifo_interface

Register Information					
Description	"External FIFO Interface Enable (only valid when gem_host_if_soft_select is defined)"				
Offset	0x04C		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:1	reserved_31_1	"Reserved, read as 0, ignored on write."		RO	0x0000 0000
0	external_fifo_interface	"Enable external fifo interface. 1: Enabled. 0: Disabled."		RW	0

axi_max_pipeline

Register Information					
Description	"Used to set the maximum amount of outstanding transactions on the AXI bus between AR / R channels and AW / W channels. Cannot be more than the depth of the configured AXI pipeline FIFO (defined in verilog defs.v)"				
Offset	0x054		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:17	reserved	"Reserved, read as 0, ignored on write."		RO	0x0000
16	use_aw2b_fill	"For the write issuing capability as defined in bits 15:8 of this register, select whether the max number of transactions operates between the AW to W AXI channel or the AW to B channel. Set to 0 to operate between the AW and W channels. Set to 1 to operate between the AW and B channels."		RW	0
15:8	aw2w_max_pipeline	"Defines the maximum number of outstanding AXI write requests that can be issued by the DMA via the AW channel. This is effectively the write issuing capability"		RW	0x01
7:0	ar2r_max_pipeline	"Defines the maximum number of outstanding AXI read requests that can be issued by the DMA via the AR channel. This is effectively the read issuing capability."		RW	0x01

rsc_control

Register Information			
Description	"Used to enable Receive side coalescing on queues 1-15"		
Offset	0x058	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:17	reserved_31_17	"Reserved, read as 0, ignored on write."	RO	0x0000
16	rsc_clr_mask	"Mask the clearing of the rsc_en bit. When set to 1 this bit will prevent the hardware from clearing the rsc_en bit when the statemachines detect a Flag set during the coalescing function."	RW	0
15:1	rsc_control	"Enables Receive Side Coalescing. Bit 1 enables RSC on queue 1, Bit 2 on queue 2 etc. RSC on queue 0 is not permitted."	RW	0x0000
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_moderation

Register Information			
Description	"Used to moderate the number of transmit and receive complete interrupts issued. With interrupt moderation enabled receive and transmit interrupts are not generated immediately a frame is transmitted or received. Instead when a receive or transmit event occurs a timer is started and the interrupt is asserted after it times out. This limits the frequency with which the CPU receives interrupts. When interrupt moderation is enabled interrupt status bit one is always used for receive and bit 7 is always used for transmit even when priority queuing is enabled. With interrupt moderation 800ns periods are counted. GEM determines what constitutes an 800ns period by looking at the tbi (bit 11), gigabit bit (10) and speed (bit 0) bits in the network configuration register and counting tx_clk cycles. Bit 0 needs to be set to 1 for 100M operation."		
Offset	0x05C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:24	reserved_31_24	"Reserved, read as 0, ignored on write."	RO	0x00
23:16	tx_int_moderation	"Count of 800ns periods before bit 7 is set in the interrupt status register after a frame is transmitted. A non-zero value indicates transmit interrupt moderation will be performed."	RW	0x00
15:8	reserved_15_8	"Reserved, read as 0, ignored on write."	RO	0x00
7:0	rx_int_moderation	"Count of 800ns periods before bit 1 is set in the interrupt status register after a frame is received. A non-zero value indicates receive interrupt moderation will be performed."	RW	0x00

sys_wake_time

Register Information			
Description	"Used to pause transmission after deassertion of tx_lpi_en. Each unit in this register corresponds to 64ns in gigabit mode, 320ns in 100M mode and 3200ns at 10M. After tx_lpi_en is deasserted transmission will pause for the set time."		
Offset	0x060	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	sys_wake_time	"Count of 25.6ns, 64ns, 320ns or 3200ns intervals before transmission starts after deassertion of tx_lpi_en (each interval is equivalent to eight tx_clk periods and so varies with data rate)."	RW	0x0000

hash_bottom

Register Information			
Description	"The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames. Hash Register Bottom 31:0"		
Offset	0x080	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"The first 32 bits of the hash address register."	RW	0x0000 0000

hash_top

Register Information			
Description	"Hash Register Top 63:32"		
Offset	0x084	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"The remaining 32 bits of the hash address register."	RW	0x0000 0000

spec_add1_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x088		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."		RW	0x0000 0000

spec_add1_top

Register Information					
Description	"Specific Address Top"				
Offset	0x08C		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:17	reserved_31_17	"Reserved, read as 0, ignored on write."		RO	0x0000
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."		RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."		RW	0x0000

spec_add2_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x090		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."		RW	0x0000 0000

spec_add2_top

Register Information					
Description	"Specific Address Top"				
Offset	0x094		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."		RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."		RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."		RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."		RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."		RW	0x0000

spec_add3_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x098		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."		RW	0x0000 0000

spec_add3_top

Register Information				
Description	"Specific Address Top"			
Offset	0x09C	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add4_bottom

Register Information				
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "			
Offset	0x0A0	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add4_top

Register Information				
Description	"Specific Address Top"			
Offset	0x0A4	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_type1

Register Information				
Description	"Type ID Match 1"			
Offset	0x0A8	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	enable_copy	"Enable copying of type ID match 1 matched frames."	RW	0
30:16	reserved_30_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	match	"Type ID match 1. For use in comparisons with received frames type ID/length field."	RW	0x0000

spec_type2

Register Information				
Description	"Type ID Match 2"			
Offset	0x0AC	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	enable_copy	"Enable copying of type ID match 2 matched frames."	RW	0
30:16	reserved_30_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	match	"Type ID match 2. For use in comparisons with received frames type ID/length field."	RW	0x0000

spec_type3

Register Information					
Description	"Type ID Match 3"				
Offset	0x0B0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31	enable_copy	"Enable copying of type ID match 3 matched frames."	RW	0	
30:16	reserved_30_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:0	match	"Type ID match 3. For use in comparisons with received frames type ID/length field."	RW	0x0000	

spec_type4

Register Information					
Description	"Type ID Match 4"				
Offset	0x0B4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31	enable_copy	"Enable copying of type ID match 4 matched frames."	RW	0	
30:16	reserved_30_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:0	match	"Type ID match 4. For use in comparisons with received frames type ID/length field."	RW	0x0000	

wol_register

Register Information					
Description	"Wake on LAN Register"				
Offset	0x0B8		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:20	reserved_31_20	"Reserved - read 0, ignored when written."		RO	0x000
19	wol_mask_3	"Wake on LAN multicast hash event enable. When set multicast hash events will cause the wol output to be asserted."		RW	0
18	wol_mask_2	"Wake on LAN specific address register 1 event enable. When set specific address 1 events will cause the wol output to be asserted."		RW	0
17	wol_mask_1	"Wake on LAN ARP request event enable. When set ARP request events will cause the wol output to be asserted."		RW	0
16	wol_mask_0	"Wake on LAN magic packet event enable. When set magic packet events will cause the wol output to be asserted."		RW	0
15:0	addr	"Wake on LAN ARP request IP address. Written to define the least significant 16 bits of the target IP address that is matched to generate a Wake on LAN event. A value of zero will not generate an event, even if this is matched by the received frame."		RW	0x0000

stretch_ratio

Register Information					
Description	"IPG stretch register"				
Offset	0x0BC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."		RO	0x0000
15:0	ipg_stretch	"IPG Stretch. Bits 7:0 are multiplied with the previously transmitted frame length (including preamble) bits 15:8 +1 divide the frame length. If the resulting number is greater than 96 and bit 28 is set in the network configuration register then the resulting number is used for the transmit inter-packet-gap. 1 is added to bits 15:8 to prevent a divide by zero."		RW	0x0000

stacked_vlan

Register Information					
Description	"Stacked VLAN Register"				
Offset	0x0C0	**Type:**	RW		
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31	enable_processing	"Enable stacked VLAN processing mode"	RW	0	
30:16	reserved_30_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:0	match	"User defined VLAN_TYPE field. When Stacked VLAN is enabled, the first VLAN tag in a received frame will only be accepted if the VLAN type field is equal to this user defined VLAN_TYPE OR equal to the standard VLAN type (0x8100). Note that the second VLAN tag of a Stacked VLAN packet will only be matched correctly if its VLAN_TYPE field equals 0x8100."	RW	0x0000	

tx_pfc_pause

Register Information					
Description	"Transmit PFC Pause Register "				
Offset	0x0C4	**Type:**	RW		
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:8	vector	"Priority Vector Pause Size. If bit 17 of the network control register is written with a one then for each entry equal to zero in the Transmit PFC Pause Register[15:8], the PFC pause frame's pause quantum field associated with that entry will be taken from the transmit pause quantum register. For each entry equal to one in the Transmit PFC Pause Register [15:8], the pause quantum associated with that entry will be zero."	RW	0x00	
7:0	vector_enable	"Priority Vector Enable. If bit 17 of the network control register is written with a one then the priority enable vector of the PFC priority based pause frame will be set equal to the value stored in this register [7:0]."	RW	0x00	

mask_add1_bottom

Register Information					
Description	"Specific Address Mask 1 Bottom 31:0"				
Offset	0x0C8	**Type:**	RW		
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:0	address_mask	"Specific Address Mask. Setting a bit to one masks the corresponding bit in the specific address 1 register"	RW	0x0000 0000	

mask_add1_top

Register Information			
Description	"Specific Address Mask 1 Top 47:32"		
Offset	0x0CC	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	address_mask	"Specific Address Mask. Setting a bit to one masks the corresponding bit in the specific address 1 register"	RW	0x0000

dma_addr_or_mask

Register Information			
Description	"Receive DMA Data Buffer Address Mask "		
Offset	0x0D0	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:28	mask_value	"Data Buffer Address Mask Value. Values used to force bits 31:28 of the receive data buffer AHB/AXI address to a particular value when the associated enable bits stored in this register [3:0] are set. Any changes to this register will be ignored while the DMA is currently processing a receive packet. It will only affect the next full packet to be written to external system memory."	RW	0x0
27:4	reserved_27_4	"Reserved, read as 0, ignored on write."	RO	0x00 0000
3:0	mask_enable	"Data Buffer Address Mask Enable. These bits are associated directly with bits[31:28].When bit 0 is set, the AHB/AXI address bit 28 used for accessing the receive data buffers will be forced to the value stored in bit 28 of this register. When bit 1 is set, the AHB/AXI address bit 29 used for accessing the receive data buffers will be forced to the value stored in bit 29 of this register. When bit 2 is set, the AHB/AXI address bit 30 used for accessing the receive data buffers will be forced to the value stored in bit 30 of this register. When bit 3 is set, the AHB/AXI address bit 31 used for accessing the receive data buffers will be forced to the value stored in bit 31 of this register. When these bits are clear, the associated value stored in bits 31:28 have no effect on the AHB/AXI address used for receive data buffer accesses. Any changes to this register will be ignored while the DMA is currently processing a receive packet. It will only affect the next full packet to be written to external memory."	RW	0x0

rx_ptp_unicast

Register Information					
Description	"PTP RX unicast IP destination address "				
Offset	0x0D4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:0	address	"Unicast IP destination address. Used for detection of PTP frames on receive path."	RW	0x0000 0000	

tx_ptp_unicast

Register Information					
Description	"PTP TX unicast IP destination address "				
Offset	0x0D8		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:0	address	"Unicast IP destination address. Used for detection of PTP frames on transmit path."	RW	0x0000 0000	

tsu_nsec_cmp

Register Information					
Description	"TSU timer comparison value nanoseconds "				
Offset	0x0DC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:22	reserved_31_22	"Reserved, read as 0, ignored on write."	RO	0x000	
21:0	comparison_value	"TSU timer comparison value (ns). Value is compared to the bits[45:24] of the TSU timer count value (upper 22 bits of nanosecond value)."	RW	0x00 0000	

tsu_sec_cmp

Register Information					
Description	"TSU timer comparison value seconds 31:0"				
Offset	0x0E0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:0	comparison_value	"TSU timer comparison value (s). Value is compared to seconds value bits [31:0] of the TSU timer count value."	RW	0x0000 0000	

tsu_msb_sec_cmp

Register Information					
Description	"TSU timer comparison value seconds 47:32"				
Offset	0x0E4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:0	comparison_value	"TSU timer comparison value (s). Value is compared to the top 16 bits (most significant 16-bits {47:32] of seconds value) of the TSU timer count value."	RW	0x0000	

tsu_ptp_tx_msb_sec

Register Information					
Description	"PTP Event Frame Transmitted Seconds Register 47:32"				
Offset	0x0E8		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:0	timer_seconds	"PTP Event Frame TX Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated."	RO	0x0000	

tsu_ptp_rx_msb_sec

Register Information					
Description	"PTP Event Frame Received Seconds Register 47:32"				
Offset	0x0EC		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:0	timer_seconds	"PTP Event Frame TX Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP receive primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated."	RO	0x0000	

tsu_peer_tx_msb_sec

Register Information					
Description	"PTP Peer Event Frame Transmitted Seconds Register 47:32"				
Offset	0x0F0		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."		RO	0x0000
15:0	timer_seconds	"PTP Peer Event Frame TX Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated."		RO	0x0000

tsu_peer_rx_msb_sec

Register Information					
Description	"PTP Peer Event Frame Received Seconds Register 47:32"				
Offset	0x0F4		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."		RO	0x0000
15:0	timer_seconds	"PTP Peer Event Frame RX Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP receive peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated."		RO	0x0000

dpram_fill_dbg

Register Information			
Description	"The fill levels for the TX & RX packet buffers can be read using this register, including the fill level for each queue in the TX direction."		
Offset	0x0F8	**Type:**	RW

Bits	Name	Description	Access	Reset
31:16	dma_tx_rx_fill_level	"Fill Level - TX or RX packet buffer fill level, selected by the tx_q_fill_level_select and tx_rx_fill_level_select registers. Read this register to determine the fill level."	RO	0x0000
15:8	reserved_15_8	"Reserved, read as 0, ignored on write."	RO	0x00
7:4	dma_tx_q_fill_level_select	"TX queue fill level select - select what TX queue to report fill levels for."	RW	0x0
3:1	reserved_3_1	"Reserved, read as 0, ignored on write."	RO	0x0
0	dma_tx_rx_fill_level_select	"TX/RX Fill Level select - report the fill level for the TX or RX packet buffer."	RW	0

revision_reg

Register Information			
Description	"This register indicates a Cadence module identification number and module revision. The value of this register is read only as defined by `gem_revision_reg_value"		
Offset	0x0FC	**Type:**	RO

Bits	Name	Description	Access	Reset
31:28	fix_number	"Fix number - incremented for fix releases."	RO	0x0
27:16	module_identification_number	"Module identification number - for the GEM, this value is fixed."	RO	0x007
15:0	module_revision	"Module revision - fixed value specific to the revision of the design which is incremented for each non-fix release of the IP."	RO	0x0100

octets_txed_bottom

Register Information			
Description	"These registers reset to zero on a read and stick at all ones when they count to their maximum value. They should be read frequently enough to prevent loss of data. In order to reduce overall design area, the statistics registers may be optionally removed in the configuration file if they are deemed unnecessary for a particular design. The receive statistics registers are only incremented when the receive enable bit is set in the network control register. The statistics registers optionally have a snapshot capability which, when exercised, will simultaneously store and clear the current values of all the statistics registers into a snapshot register set in order to allow a consistent set of statistics to be read by the processor. The snapshot is controlled using bit 13 of the network control register. The read snapshot control indicated by bit 14 of the network control register determines whether the processor reads the snapshot registers (logic 1) or the incrementing registers (logic 0). The default GEM configuration does not support the snapshot capability. See Parameterization section under Implementation Application Notes for an explanation of how to enable this function. All the statistics registers are read only. For test purposes they may be written by setting bit 7 (Write Enable) in the network control register. Setting bit 6 (increment statistics) in the network control register causes all the statistics registers to increment by one, again for test purposes. Once a statistics register has been read, it is automatically cleared. When reading the octets transmitted and octets received registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation. The statistics register block contains the following registers. Octets Transmitted [31:0]"		
Offset	0x100	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"Transmitted octets in frame without errors [31:0]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames."	RO RtoClr	0x0000 0000

octets_txed_top

Register Information			
Description	"Octets Transmitted 47:32"		
Offset	0x104	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	count	"Transmitted octets in frame without errors [47:32]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames."	RO RtoClr	0x0000

frames_txed_ok

Register Information					
Description	"Frames Transmitted"				
Offset	0x108		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:0	count	"Frames transmitted without error. A 32 bit register counting the number of frames successfully transmitted, i.e. no under run and not too many retries. Excludes pause frames."	RO RtoClr	0x0000 0000	

broadcast_txed

Register Information					
Description	"Broadcast Frames Transmitted"				
Offset	0x10C		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:0	count	"Broadcast frames transmitted without error. A 32 bit register counting the number of broadcast frames successfully transmitted without error, i.e. no under run and not too many retries. Excludes pause frames."	RO RtoClr	0x0000 0000	

multicast_txed

Register Information					
Description	"Multicast Frames Transmitted"				
Offset	0x110		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:0	count	"Multicast frames transmitted without error. A 32 bit register counting the number of multicast frames successfully transmitted without error, i.e. no under run and not too many retries. Excludes pause frames."	RO RtoClr	0x0000 0000	

pause_frames_txed

Register Information			
Description	"Pause Frames Transmitted"		
Offset	0x114	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	count	"Transmitted pause frames - a 16 bit register counting the number of pause frames transmitted. Only pause frames triggered by the register interface or through the external pause pins are counted as pause frames. Pause frames received through the external FIFO interface are counted in the frames transmitted counter."	RO RtoClr	0x0000

frames_txed_64

Register Information			
Description	"64 Byte Frames Transmitted"		
Offset	0x118	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"64 byte frames transmitted without error. A 32 bit register counting the number of 64 byte frames successfully transmitted without error, i.e. no under run and not too many retries. Excludes pause frames."	RO RtoClr	0x0000 0000

frames_txed_65

Register Information			
Description	"65 to 127 Byte Frames Transmitted"		
Offset	0x11C	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"65 to127 byte frames transmitted without error. A 32 bit register counting the number of 65 to127 byte frames successfully transmitted without error, i.e. no under run and not too many retries."	RO RtoClr	0x0000 0000

frames_txed_128

Register Information					
Description	"128 to 255 Byte Frames Transmitted"				
Offset	0x120		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	count	"128 to 255 byte frames transmitted without error. A 32 bit register counting the number of 128 to 255 byte frames successfully transmitted without error, i.e. no under run and not too many retries."		RO RtoClr	0x0000 0000

frames_txed_256

Register Information					
Description	"256 to 511 Byte Frames Transmitted"				
Offset	0x124		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	count	"256 to 511 byte frames transmitted without error. A 32 bit register counting the number of 256 to 511 byte frames successfully transmitted without error, i.e. no under run and not too many retries."		RO RtoClr	0x0000 0000

frames_txed_512

Register Information					
Description	"512 to 1023 Byte Frames Transmitted"				
Offset	0x128		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	count	"512 to 1023 byte frames transmitted without error. A 32 bit register counting the number of 512 to 1023 byte frames successfully transmitted without error, i.e. no under run and not too many retries."		RO RtoClr	0x0000 0000

frames_txed_1024

Register Information					
Description	"1024 to 1518 Byte Frames Transmitted"				
Offset	0x12C		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	count	"1024 to 1518 byte frames transmitted without error. A 32 bit register counting the number of 1024 to 1518 byte frames successfully transmitted without error, i.e. no under run and not too many retries."		RO RtoClr	0x0000 0000

frames_txed_1519

Register Information					
Description	"Greater Than 1518 Byte Frames Transmitted"				
Offset	0x130		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	count	"Greater than 1518 byte frames transmitted without error. A 32 bit register counting the number of 1518 or above byte frames successfully transmitted without error, i.e. no under run and not too many retries."		RO RtoClr	0x0000 0000

tx_underruns

Register Information					
Description	"Transmit Under Runs"				
Offset	0x134		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_22	"Reserved, read as 0, ignored on write."		RO	0x00 0000
9:0	count	"Transmit under runs - a 10 bit register counting the number of frames not transmitted due to a transmit under run. If this register is incremented then no other statistics register is incremented."		RO RtoClr	0x000

single_collisions

Register Information					
Description	"Single Collision Frames"				
Offset	0x138		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:18	reserved_31_18	"Reserved, read as 0, ignored on write."		RO	0x0000
17:0	count	"Single collision frames - an 18 bit register counting the number of frames experiencing a single collision before being successfully transmitted, i.e. no under run."		RO RtoClr	0x0 0000

multiple_collisions

Register Information					
Description	"Multiple Collision Frames"				
Offset	0x13C		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:18	reserved_31_18	"Reserved, read as 0, ignored on write."		RO	0x0000
17:0	count	"Multiple collision frames - an 18 bit register counting the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e. no under run and not too many retries."		RO RtoClr	0x0 0000

excessive_collisions

Register Information					
Description	"Excessive Collisions"				
Offset	0x140		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."		RO	0x00 0000
9:0	count	"Excessive collisions - a 10 bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions."		RO RtoClr	0x000

late_collisions

Register Information					
Description	"Late Collisions"				
Offset	0x144		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."		RO	0x00 0000
9:0	count	"Late collisions - a 10 bit register counting the number of late collision occurring after the slot time (512 bits) has expired. In 10/100 mode, late collisions are counted twice i.e. both as a collision and a late collision. In gigabit mode, a late collision causes the transmission to be aborted, thus the single and multi collision registers are not updated."		RO RtoClr	0x000

deferred_frames

Register Information				
Description	"Deferred Transmission Frames"			
Offset	0x148	**Type:**	RO	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:18	reserved_31_18	"Reserved, read as 0, ignored on write."	RO	0x0000
17:0	count	"Deferred transmission frames - an 18 bit register counting the number of frames experiencing deferral due to carrier sense being active on their first attempt at transmission. Frames involved in any collision are not counted nor are frames that experienced a transmit under run."	RO RtoClr	0x0 0000

crs_errors

Register Information				
Description	"Carrier Sense Errors"			
Offset	0x14C	**Type:**	RO	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9:0	count	"Carrier sense errors - a 10 bit register counting the number of frames transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit frame without collision (no under run). Only incremented in half duplex mode. The only effect of a carrier sense error is to increment this register. The behaviour of the other statistics registers is unaffected by the detection of a carrier sense error."	RO RtoClr	0x000

octets_rxed_bottom

Register Information				
Description	"Octets Received 31:0"			
Offset	0x150	**Type:**	RO	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"Received octets in frame without errors [31:0]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

octets_rxed_top

Register Information				
Description	"Octets Received 47:32"			
Offset	0x154	**Type:**	RO	

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	count	"Received octets in frame without errors [47:32]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000

frames_rxed_ok

Register Information				
Description	"Frames Received"			
Offset	0x158	**Type:**	RO	

Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"Frames received without error. A 32 bit register counting the number of frames successfully received. Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

broadcast_rxed

Register Information				
Description	"Broadcast Frames Received"			
Offset	0x15C	**Type:**	RO	

Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"Broadcast frames received without error. A 32 bit register counting the number of broadcast frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

multicast_rxed

Register Information					
Description	"Multicast Frames Received"				
Offset	0x160		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	count	"Multicast frames received without error. A 32 bit register counting the number of multicast frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."		RO RtoClr	0x0000 0000

pause_frames_rxed

Register Information					
Description	"Pause Frames Received"				
Offset	0x164		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."		RO	0x0000
15:0	count	"Received pause frames - a 16 bit register counting the number of pause frames received without error."		RO RtoClr	0x0000

frames_rxed_64

Register Information					
Description	"64 Byte Frames Received"				
Offset	0x168		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	count	"64 byte frames received without error. A 32 bit register counting the number of 64 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."		RO RtoClr	0x0000 0000

frames_rxed_65

Register Information			
Description	"65 to 127 Byte Frames Received"		
Offset	0x16C	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"65 to 127 byte frames received without error. A 32 bit register counting the number of 65 to 127 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

frames_rxed_128

Register Information			
Description	"128 to 255 Byte Frames Received"		
Offset	0x170	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"128 to 255 byte frames received without error. A 32 bit register counting the number of 128 to 255 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

frames_rxed_256

Register Information			
Description	"256 to 511 Byte Frames Received"		
Offset	0x174	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"256 to 511 byte frames received without error. A 32 bit register counting the number of 256 to 511 byte frames successfully transmitted without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

frames_rxed_512

Register Information			
Description	"512 to 1023 Byte Frames Received"		
Offset	0x178	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"512 to 1023 byte frames received without error. A 32 bit register counting the number of 512 to 1023 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

frames_rxed_1024

Register Information			
Description	"1024 to 1518 Byte Frames Received"		
Offset	0x17C	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"1024 to 1518 byte frames received without error. A 32 bit register counting the number of 1024 to 1518 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

frames_rxed_1519

Register Information			
Description	"1519 to maximum Byte Frames Received"		
Offset	0x180	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	count	"1519 to maximum byte frames received without error. A 32 bit register counting the number of 1519 byte or above frames successfully received without error. Maximum frame size is determined by the network configuration register bit 8 (1536 maximum frame size) or bit 3 (jumbo frame size). Excludes pause frames, and is only incremented if the frame is successfully filtered."	RO RtoClr	0x0000 0000

undersize_frames

Register Information					
Description	"Undersized Frames Received"				
Offset	0x184		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."		RO	0x00 0000
9:0	count	"Undersize frames received - a 10 bit register counting the number of frames received less than 64 bytes in length (10/100 mode or gigabit mode, full duplex) that do not have either a CRC error or an alignment error. In gigabit mode, half duplex, this register counts either frames not conforming to the minimum slot time of 512 bytes or frames not conforming to the minimum frame size once bursting is active."		RO RtoClr	0x000

excessive_rx_length

Register Information					
Description	"Oversize Frames Received"				
Offset	0x188		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_31_10	"Reserved read 0, ignored on write."		RO	0x00 0000
9:0	count	"Oversize frames received - a 10 bit register counting the number of frames received exceeding 1518 bytes (1536 bytes if bit 8 is set in network configuration register, 10,240 bytes if bit 3 is set in the network configuration register) in length but do not have either a CRC error, an alignment error nor a receive symbol error."		RO RtoClr	0x000

rx_jabbers

Register Information					
Description	"Jabbers Received"				
Offset	0x18C		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."		RO	0x00 0000
9:0	count	"Jabbers received - a 10 bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register, 10,240 bytes if bit 3 is set in the network configuration register) in length and have either a CRC error, an alignment error or a receive symbol error."		RO RtoClr	0x000

fcs_errors

Register Information				
Description	"Frame Check Sequence Errors"			
Offset	0x190	**Type:**	RO	

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9:0	count	"Frame check sequence errors - a 10 bit register counting frames that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register, 10,240 bytes if bit 3 is set in the network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and has an integral number of bytes. This register is incremented for a frame with bad FCS, regardless of whether it is copied to memory due to ignore FCS mode being enabled in bit 26 of the network configuration register."	RO RtoClr	0x000

rx_length_errors

Register Information				
Description	"Length Field Frame Errors"			
Offset	0x194	**Type:**	RO	

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9:0	count	"Length field frame errors - this 10-bit register counts the number of frames received that have a measured length shorter than that extracted from the length field (bytes 13 and 14). This condition is only counted if the value of the length field is less than 0x0600, the frame is not of excessive length and checking is enabled through bit 16 of the network configuration register."	RO RtoClr	0x000

rx_symbol_errors

Register Information					
Description	"Receive Symbol Errors"				
Offset	0x198		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."		RO	0x00 0000
9:0	count	"Receive symbol errors - a 10-bit register counting the number of frames that had rx_er asserted during reception. For 10/100 mode symbol errors are counted regardless of frame length checks. For gigabit mode the frame must satisfy slot time requirements in order to count a symbol error. Additionally, in gigabit half duplex mode, carrier extension errors are also recorded. Receive symbol errors will also be counted as an FCS or alignment error if the frame is between 64 and 1518 bytes (1536 bytes if bit 8 is set in the network configuration register, 10240 bytes if bit 3 is set in the network configuration register). If the frame is larger it will be recorded as a jabber error."		RO RtoClr	0x000

alignment_errors

Register Information					
Description	"Alignment Errors"				
Offset	0x19C		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."		RO	0x00 0000
9:0	count	"Alignment errors - a 10 bit register counting frames that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register, 10,240 bytes if bit 3 is set in the network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes."		RO RtoClr	0x000

rx_resource_errors

Register Information			
Description	"Receive Resource Errors"		
Offset	0x1A0	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:18	reserved_31_18	"Reserved, read as 0, ignored on write."	RO	0x0000
17:0	count	"Receive resource errors - an 18 bit register counting the number of frames that were successfully received by the MAC (correct address matched frame and adequate slot time) but could not be copied to memory because no receive buffer was available. This occurs when the GEM reads a buffer descriptor with its ownership (or used) bit set."	RO RtoClr	0x0 0000

rx_overruns

Register Information			
Description	"Receive Overruns"		
Offset	0x1A4	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9:0	count	"Receive overruns - a 10 bit register counting the number of frames that are address recognized but were not copied to memory due to a receive overrun."	RO RtoClr	0x000

rx_ip_ck_errors

Register Information			
Description	"IP Header Checksum Errors"		
Offset	0x1A8	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	count	"IP header checksum errors - an 8-bit register counting the number of frames discarded due to an incorrect IP header checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the network configuration register or 10240 bytes if bit 3 is in the network configuration register) and do not have a CRC error, an alignment error, nor a symbol error."	RO RtoClr	0x00

rx_tcp_ck_errors

Register Information					
Description	"TCP Checksum Errors"				
Offset	0x1AC		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."		RO	0x00 0000
7:0	count	"TCP checksum errors - an 8-bit register counting the number of frames discarded due to an incorrect TCP checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the network configuration register or 10240 bytes if bit 3 is in the network configuration register) and do not have a CRC error, an alignment error, nor a symbol error."		RO RtoClr	0x00

rx_udp_ck_errors

Register Information					
Description	"UDP Checksum Errors"				
Offset	0x1B0		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."		RO	0x00 0000
7:0	count	"UDP checksum errors - an 8-bit register counting the number of frames discarded due to an incorrect UDP checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the network configuration register or 10240 bytes if bit 3 is in the network configuration register) and do not have a CRC error, an alignment error, nor a symbol error."		RO RtoClr	0x00

auto_flushed_pkts

Register Information					
Description	"Receive DMA Flushed Packets"				
Offset	0x1B4		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000	
15:0	count	"Flushed RX pkts counter. A 16 bit register counting the number of frames that have been flushed from the receive SRAM based packet buffer due to one of the following reasons .1. When partial store and forward mode is enabled or bit 24 of the DMA configuration register is enabled, a packet is received while there is no AMBA (AHB/AXI) resource. 2. When partial store and forward mode is enabled and an AMBA (AHB/AXI) error is encountered while writing the packet data to external memory. When bit 18 of the network control register(software action to flush a packet from the head of the PBUF queue) is pulsed and the GEM DMA is not currently busy."	RO RtoClr	0x0000	

tsu_timer_incr_sub_nsec

Register Information					
Description	"1588 Timer Increment Register sub nsec. From release 1p08f1 onwards this register must be written before the tsu_timer_incr register and the value written will not take effect until the tsu_timer_incr register is written to."				
Offset	0x1BC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:24	sub_ns_incr_lsb	"These are the least significant bits [7:0] of the sub-ns value by which the 1588 timer will be incremented each clock cycle."	RW	0x00	
23:16	reserved_23_16	"Reserved, read as 0, ignored on write."	RO	0x00	
15:0	sub_ns_incr	"These are the most significant bits [23:8] of the sub-ns value by which the 1588 timer will be incremented each clock cycle. 24 bits of sub nanosecond precision gives resolution of approximately 5.86E-17 seconds (16 bits gives 15.2 femtoseconds)."	RW	0x0000	

tsu_timer_msb_sec

Register Information				
Description	"1588 Timer Seconds Register 47:32"			
Offset	0x1C0	**Type:**	RW	

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	timer	"TSU timer value (s). Most significant 16 bits of seconds timer count. The register is writeable. The 48-bit counter increments by one when the 1588 nanoseconds counter counts to one second. It may also be incremented or decremented when the timer adjust register is written (if decremented from zero the 48-bit combined count would roll back to 0xFFFFFFFFFFFF). Note: The value of this register is used only when the lower 32 bit register is written to. This is to ensure a single update of the 48 bit seconds value"	RW	0x0000

tsu_strobe_msb_sec

Register Information				
Description	"1588 Timer Sync Strobe Seconds Register 47:32"			
Offset	0x1C4	**Type:**	RO	

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	reserved_31_16	"Reserved, read as 0, ignored on write."	RO	0x0000
15:0	strobe	"1588 Timer Sync Strobe Seconds. The most significant 16-bit value of the Timer Seconds register captured when gem_tsu_ms and gem_tsu_inc_ctrl are zero."	RO	0x0000

tsu_strobe_sec

Register Information				
Description	"1588 Timer Sync Strobe Seconds Register 31:0"			
Offset	0x1C8	**Type:**	RO	

Bitfield Details

Bits	Name	Description	Access	Reset
31:0	strobe	"1588 Timer Sync Strobe Seconds. The lowest significant 32-bit value of the Timer Seconds register captured when gem_tsu_ms and gem_tsu_inc_ctrl are zero."	RO	0x0000 0000

tsu_strobe_nsec

Register Information					
Description	"1588 Timer Sync Strobe Nanoseconds Register"				
Offset	0x1CC		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."		RO	0x0
29:0	strobe	"1588 Timer Sync Strobe Nanoseconds. The value of the Timer Nanoseconds register captured when gem_tsu_ms and gem_tsu_inc_ctrl are zero."		RO	0x0000 0000

tsu_timer_sec

Register Information					
Description	"1588 Timer Seconds Register 31:0"				
Offset	0x1D0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	timer	"1588 Timer Seconds Register. TSU timer value (s). Least significant 32 bits of seconds timer count. This register is writeable. The 48-bit counter increments by one when the 1588 nanoseconds counter counts to one second. It may also be incremented or decremented when the timer adjust register is written (if decremented from zero the 48-bit combined count would roll back to 0xFFFFFFFFFFFF)."		RW	0x0000 0000

tsu_timer_nsec

Register Information					
Description	"1588 Timer Nanoseconds Register"				
Offset	0x1D4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."		RO	0x0
29:0	timer	"Timer count in nanoseconds. This register is writeable. It can also be adjusted by writes to the 1588 timer adjust register. It increments by the value of the 1588 timer increment register each clock cycle (if this register is close to zero and a write to the timer adjust register causes a decrement the seconds register will be decremented if necessary and the nanoseconds register will roll back to 9999999xx(dec))."		RW	0x0000 0000

tsu_timer_adjust

Register Information					
Description	"This register returns all zeroes when read."				
Offset	0x1D8		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31	add_subtract	"Write as one to subtract from the 1588 timer. Write as zero to add to it."	WO	0	
30	reserved_30_30	"Reserved, read as 0, ignored on write."	RO	0	
29:0	increment_value	"Timer increment value. The number of nanoseconds to increment or decrement the 1588 timer nanoseconds register. If necessary the 1588 seconds register will be incremented or decremented."	WO	0x0000 0000	

tsu_timer_incr

Register Information					
Description	"1588 Timer Increment Register. From release 1p08f1 onwards this register must be written after the tsu_timer_incr_sub_ns register and the write operation will cause the value written to the tsu_timer_incr_sub_ns register to take effect."				
Offset	0x1DC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:24	reserved_31_24	"Reserved, read as 0, ignored on write."	RO	0x00	
23:16	num_incs	"Number of incs before alt inc. The number of increments after which the alternative increment is used."	RW	0x00	
15:8	alt_ns_incr	"Alternative nanoseconds count. Alternative count of nanoseconds by which the 1588 timer nanoseconds register will be incremented each clock cycle."	RW	0x00	
7:0	ns_increment	"A count of nanoseconds by which the 1588 timer nanoseconds register will be incremented each clock cycle. These are the most significant 8 bits of the 32 bit timer_increment counter. The tsu_timer_incr_sub_nsec register holds the least significant 24 bits of the increment."	RW	0x00	

tsu_ptp_tx_sec

Register Information					
Description	"PTP Event Frame Transmitted Seconds Register 31:0"				
Offset	0x1E0		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	timer	"PTP Event Frame Transmitted Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated."		RO	0x0000 0000

tsu_ptp_tx_nsec

Register Information					
Description	"PTP Event Frame Transmitted Nanoseconds Register"				
Offset	0x1E4		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."		RO	0x0
29:0	timer	"PTP Event Frame Transmitted Nanoseconds. The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP transmit primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated."		RO	0x0000 0000

tsu_ptp_rx_sec

Register Information					
Description	"PTP Event Frame Received Seconds Register 31:0"				
Offset	0x1E8		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	timer	"PTP Event Frame Received Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP receive primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated."		RO	0x0000 0000

tsu_ptp_rx_nsec

Register Information			
Description	"PTP Event Frame Received Nanoseconds Register"		
Offset	0x1EC	**Type:**	RO

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:0	timer	"PTP Event Frame Received Nanoseconds. The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP receive primary event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP sync or delay_req frame. An interrupt is issued when the register is updated."	RO	0x0000 0000

tsu_peer_tx_sec

Register Information			
Description	"PTP Peer Event Frame Transmitted Seconds Register 31:0"		
Offset	0x1F0	**Type:**	RO

Bits	Name	Description	Access	Reset
31:0	timer	"PTP Peer Event Frame Received Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated."	RO	0x0000 0000

tsu_peer_tx_nsec

Register Information			
Description	"PTP Peer Event Frame Transmitted Nanoseconds Register"		
Offset	0x1F4	**Type:**	RO

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:0	timer	"PTP Peer Event Frame Transmitted Nanoseconds. The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP transmit peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated."	RO	0x0000 0000

tsu_peer_rx_sec

Register Information					
Description	"PTP Peer Event Frame Received Seconds Register 31:0"				
Offset	0x1F8		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	timer	"PTP Peer Event Frame Received Seconds. The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP receive peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated."		RO	0x0000 0000

tsu_peer_rx_nsec

Register Information					
Description	"PTP Peer Event Frame Received Nanoseconds Register"				
Offset	0x1FC		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."		RO	0x0
29:0	timer	"PTP Peer Event Frame Received Nanoseconds. The register is updated with the value that the 1588 timer nanoseconds register held when the SFD of a PTP receive peer event crosses the MII interface. The actual update occurs when the GEM recognizes the frame as a PTP pdelay_req or pdelay_resp frame. An interrupt is issued when the register is updated."		RO	0x0000 0000

pcs_control

Register Information			
Description	"Note: All PCS registers are defined in the IEEE 802.3 Standard. PCS Control Register. This register provides the main control functions with respect to the PCS."		
Offset	0x200	**Type:**	RW

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved. Set to zero."	RO	0x0000
15	pcs_software_reset	"PCS software reset - written by software to force the hardware logic into a reset state. This bit is self clearing. When reading this bit, logic 1 is returned until both the soft reset has completed and the PCS is enabled through the PCS select bit of the network configuration register. Writing logic 0 has no affect."	RW	1
14	loopback_mode	"Loopback mode - the ewrap output pin of the GEM reflects this control bit, and can be used to select loopback mode in the PHY transceiver. 0: Loopback mode disabled. 1: Loopback mode enabled."	RW	0
13	speed_select_bit_1	"Speed select bit 1 - combined with speed select [0] to indicate the speed of operation of the PCS. As the GEM PCS is only intended to operate at 1000 Mbps, this bit is hardwired to logic 0."	RO	0
12	enable_auto_neg	"Enable auto-negotiation - when set active high, auto-negotiation operation is enabled."	RW	1
11:10	reserved_11_10	"Reserved. Set to zero."	RO	0x0
9	restart_auto_neg	"Restart auto-negotiation - when set active high, the hardware restarts auto-negotiation. This bit is self clearing, but once set shall remain in this state until auto-negotiation has restarted. Writing logic 0 has no affect."	RW	0
8	mac_duplex_state	"MAC Duplex state. This returns the value of the MAC's duplex state as indicated in bit 1 of the MAC's network configuration register."	RO	0
7	collision_test	"Collision test - when set active high, the PCS generates collisions on transmit. This bit should only be set for test purposes."	RW	0
6	speed_select_bit_0	"Speed select bit 0 - combined with speed select [1] to indicate the speed of operation of the PCS. As the GEM PCS is only intended to operate at 1000 Mbps, this bit is hardwired to logic 1."	RO	1
5:0	reserved_5_0	"Reserved. Set to zero."	RO	0x00

pcs_status

Register Information			
Description	"This register indicates general status information concerning the PCS."		
Offset	0x204	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	reserved_31_16	"Reserved. Set to zero."	RO	0x0000
15	base_100_t4	"100 BASE-T4 - the GEM PCS does not support 100 BASE-T4. This bit is hardwired to logic 0."	RO	0
14	base_100_x_full_duplex	"100 BASE-X full duplex - the GEM PCS does not support 100 BASE-X. This bit is hardwired to logic 0."	RO	0
13	base_100_x_half_duplex	"100 BASE-X half duplex - the GEM PCS does not support 100 BASE-X. This bit is hardwired to logic 0."	RO	0
12	mbps_10_full_duplex	"10 Mbps full duplex - the GEM PCS does not support this mode. This bit is hardwired to logic 0."	RO	0
11	mbps_10_half_duplex	"10 Mbps half duplex - the GEM PCS does not support this mode. This bit is hardwired to logic 0."	RO	0
10	base_100_t2_full_duplex	"100 BASE-T2 full duplex - the GEM PCS does not support 100 BASE-T2. This bit is hardwired to logic 0."	RO	0
9	base_100_t2_half_duplex	"100 BASE-T2 half duplex - the GEM PCS does not support 100 BASE-T2. This bit is hardwired to logic 0."	RO	0
8	extended_status	"Extended status - when set active high, indicates extended status information is present in the PCS auto-negotiation extended status register. This bit is hardwired to logic 1."	RO	1
7:6	reserved_7_6	"Reserved. Set to zero."	RO	0x0
5	auto_neg_complete	"Auto-negotiation complete - set active high by the PCS hardware to indicate auto-negotiation has completed."	RO	0
4	remote_fault	"Remote fault - set active high if the link partner remote fault bits in the PCS auto-negotiation link partner ability register, indicates an error. Resets low when read."	RO	0
3	auto_neg_ability	"Auto-negotiation ability - this bit indicates whether the PCS has auto-negotiation ability and reflects the value of the auto-negotiation enable bit in the PCS control register. 0: PCS is not able to perform auto-negotiation. 1: PCS is able to perform auto-negotiation."	RO	1
2	link_status	"Link status - indicates the status of the physical connection to the link partner. When set to logic 1 the link is up, and when set to logic 0, the link is down. If auto-negotiation is disabled this returns the synchronisation status. Held at logic 0 if the link goes down until this bit is read."	RO	0
1	reserved_1	"Reserved. Set to zero."	RO	0
0	extended_capabilities	"Extended register capabilities - when set active high, indicates the PCS supports extended register capabilities. This bit is hardwired to logic 1."	RO	1

pcs_an_adv

Register Information			
Description	"The value of this register is used to transmit the base page of the GEM PCS advertised capabilities as long as SGMII mode is not enabled. If SGMII mode is enabled by setting bit 27 in the network configuration register then this register becomes read only with a fixed value of 0x00000001. (SGMII specifies that the transmit configuration information sent from the MAC to the PHY is fixed with bit 14 set to 1 to indicate acknowledge, bit 0 set to 1 to indicate SGMII and all other bits set to 0.)"		
Offset	0x210	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	reserved_31_16	"Reserved. Set to zero."	RO	0x0000
15	next_page	"Next page. When set active high, this bit is used during auto-negotiation to indicate to the link partner that the PCS requires exchanging next pages."	RW	0
14	reserved_14	"Reserved. Set to zero."	RO	0
13:12	remote_fault	"Remote fault [1:0] - indicates and classifies a remote fault condition to the link partner: 00: No error, Link O.K. 01: Link Failure. 10: Off line. 11: Auto-negotiation error."	RW	0x0
11:9	reserved_11_9	"Reserved. Set to zero."	RO	0x0
8:7	pause	"Pause[1:0] - used to provide a pause capability mechanism as follows: 00: No pause. 01: Symmetric pause. 10: Asymmetric pause toward link partner. 11: Both symmetric pause and asymmetric pause toward link device."	RW	0x0
6	half_duplex	"Half duplex - this bit defines to the link partner whether the GEM is capable of supporting half duplex operation. 0: The GEM cannot support half duplex. 1: The GEM can support half duplex."	RW	0
5	full_duplex	"Full duplex - this bit defines to the link partner whether the GEM is capable of supporting full duplex operation. 0: The GEM cannot support full duplex. 1: The GEM can support full duplex."	RW	1
4:0	reserved_4_0	"Reserved. Set to zero."	RO	0x00

pcs_an_lp_base

Register Information			
Description	"When SGMII mode is not enabled, the value of this register contains the link partner's base page received information. This register is updated in the ABILITY_DETECT state of the PCS auto-negotiation state machine so bit 14 will only be set if the link partner is sending acknowledge while the PCS in this state. The register is not updated in the ACK_DETECT state. For SGMII mode, the contents of this register change to the one defined in the SGMII standard. The value of this register contains the link partner's base page received information. In this case the link partner is the PHY connected by the SGMII."		
Offset	0x214	**Type:**	RO

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved. Set to zero."	RO	0x0000
15	link_partner_next_page_status	"The contents of this register change depending on SGMII or non SGMII mode. non SGMII Mode: Link partner next page - when set active high, this bit indicates the link partner's intention to exchange next pages. SGMII Mode: Link Status. 0 : Link Down. 1 : Link Up."	RO	0
14	link_partner_acknowledge	"Link partner acknowledge - indicates the link partner has successfully received the transmitted base page"	RO	0
13:12	link_partner_remote_fault_duplex_mode	"The contents of this register change depending on SGMII or non SGMII mode. non SGMII Mode: Link partner remote fault [1:0] - indicates and classifies a remote fault condition has been detected by the link partner as follows: 00: No error, Link O.K. 01: Link Failure. 10: Off line. 11: Auto-negotiation error. SGMII Mode: Bit 13: Reserved. read as 0. Bit 12 : 0 : Half Duplex. 1: Full Duplex."	RO	0x0
11:9	speed_reserved	"The contents of this register change depending on SGMII or non SGMII mode. non SGMII Mode: Reserved. Set to zero. SGMII Mode : Bits 11:10 : Speed : 11 : Reserved 10 : 1000 Mbps 01 : 100Mbps 00 : 10 Mbps Bit 9 : Reserved. read as 0."	RO	0x0

8:7	pause	"The contents of this register change depending on SGMII or non SGMII mode. non SGMII Mode: Pause[1:0] - provides the link partner's pause frame capability as follows: 00: No pause. 01: Symmetric pause. 10: Asymmetric pause toward link partner. 11: Both symmetric pause and asymmetric pause toward link device. SGMII Mode: Reserved. read as 0."	RO	0x0
6	link_partner_half_duplex	"The contents of this register change depending on SGMII or non SGMII mode. non SGMII Mode: Link partner half duplex - this bit indicates whether the link partner is capable of supporting half duplex operation. 0: The link partner cannot support half duplex. 1: The link partner can support half duplex. SGMII Mode: Reserved. read as 0."	RO	0
5	link_partner_full_duplex	"The contents of this register change depending on SGMII or non SGMII mode. non SGMII Mode: Link partner full duplex - this bit indicates whether the link partner is capable of supporting full duplex operation. 0: The link partner cannot support full duplex. 1: The link partner can support full duplex. SGMII Mode: Reserved. read as 0."	RO	0
4:0	reserved_4_0	"Reserved. Set to zero."	RO	0x00

pcs_an_exp

Register Information			
Description	"This register contains auto-negotiation next page ability and page received information."		
Offset	0x218	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:3	reserved_31_3	"Reserved. Set to zero."	RO	0x0000 0000
2	next_page_capability	"Next page capability - hard wired to logic 1 to indicate that the GEM PCS supports next page operation."	RO	1
1	page_received	"Page received - this bit is set active high when a new page has been received from the link partner. It is cleared when the link partner next page register has been read."	RO	0
0	reserved_0	"Reserved. Set to zero."	RO	0

pcs_an_np_tx

Register Information			
Description	"The value of this register is used to transmit the next page information for the GEM PCS. For next page exchange to work this register must be written within 10 ms of receiving a new page from the link partner. If the link partner is requesting next pages and the GEM has none or no more to send then this register should be written with the null message (0x2001). The value 0x0000 must not be written to this register."		
Offset	0x21C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	reserved_31_16	"Reserved. Set to zero."	RO	0x0000
15	next_page_to_transmit	"Next page to transmit - this bit indicates whether this is the last next page to be transmitted:0: Last page. 1: Additional page(s) to follow."	RW	0
14	reserved_14	"Reserved. Set to zero."	RO	0
13	message_page_indicator	"Message page indicator - this bit identifies the message. 0: Unformatted page. 1: Message page."	RW	0
12	acknowledge_2	"Acknowledge 2 - when set active high, indicates that the GEM PCS has the ability to comply with the last received message."	RW	0
11	reserved_11	"Reserved. Set to zero."	RO	0
10:0	message	"Message - contains data as defined by the message page indicator bit."	RW	0x000

pcs_an_lp_np

Register Information			
Description	"This value of this register contains the next page received information from the link partner."		
Offset	0x220	**Type:**	RO

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved. Set to zero."	RO	0x0000
15	next_page_to_receive	"Next page to receive - this bit indicates whether this is the last page to be received in the sequence by the GEM PCS: 0: Last page. 1: Additional page(s) to follow."	RO	0
14	acknowledge	"Acknowledge - this bit indicates whether as part of the next page function, the link partner has successfully received the last message transmitted."	RO	0
13	message_page_indicator	"Message page indicator - this bit identifies the message. 0: Unformatted page. 1: Message page."	RO	0
12	acknowledge_2	"Acknowledge 2 - set active high by the link partner to indicate when it has the ability to comply with the last message received."	RO	0
11	toggle	"Toggle - this bit toggles with every received page."	RO	0
10:0	message	"Message - contains data as defined by the message page indicator bit."	RO	0x000

pcs_an_ext_status

Register Information			
Description	"This register contains PCS auto-negotiation extended status information."		
Offset	0x23C	**Type:**	RO

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved. Set to zero."	RO	0x0000
15	full_duplex_1000base_x	"Full duplex 1000BASE-X - hardwired to logic 1, indicates the GEM PCS can support full duplex operation of 1000BASE-X."	RO	1
14	half_duplex_1000base_x	"Half duplex 1000BASE-X - hardwired to logic 0, indicates the GEM MAC cannot support half duplex operation at gigabit speeds."	RO	0
13	full_duplex_1000base_t	"Full duplex 1000BASE-T - hardwired to logic 0, indicates the GEM PCS cannot support 1000BASE-T full duplex operation."	RO	0
12	half_duplex_1000base_t	"Half duplex 1000BASE-T - hardwired to logic 0, indicates the GEM PCS cannot support 1000BASE-T half duplex operation."	RO	0
11:0	reserved_11_0	"Reserved. Set to zero."	RO	0x000

tx_pause_quantum1

Register Information				
Description	"Transmit Pause Quantum Register 1"			
Offset	0x260	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	quantum_p3	"Transmit pause quantum - written with the pause quantum value for pause frame transmission of priority 3."	RW	0xFFFF
15:0	quantum_p2	"Transmit pause quantum - written with the pause quantum value for pause frame transmission of priority 2."	RW	0xFFFF

tx_pause_quantum2

Register Information				
Description	"Transmit Pause Quantum Register 2"			
Offset	0x264	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	quantum_p5	"Transmit pause quantum - written with the pause quantum value for pause frame transmission of priority 5."	RW	0xFFFF
15:0	quantum_p4	"Transmit pause quantum - written with the pause quantum value for pause frame transmission of priority 4."	RW	0xFFFF

tx_pause_quantum3

Register Information				
Description	"Transmit Pause Quantum Register 3"			
Offset	0x268	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	quantum_p7	"Transmit pause quantum - written with the pause quantum value for pause frame transmission of priority 7."	RW	0xFFFF
15:0	quantum_p6	"Transmit pause quantum - written with the pause quantum value for pause frame transmission of priority 6."	RW	0xFFFF

rx_lpi

Register Information					
Description	"Received LPI transitions"				
Offset	0x270		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:16	reserved_31_16	"Unused, read zero"	RO	0x0000	
15:0	count	"Count of RX LPI transitions. A count of the number of times there is a transition from receiving normal idle to receiving low power idle. Cleared on read."	RO RtoClr	0x0000	

rx_lpi_time

Register Information					
Description	"Received LPI time"				
Offset	0x274		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:24	reserved_31_24	"Unused, read zero"	RO	0x00	
23:0	lpi_time	"Time in LPI. This register increments once every 16 pclk cycles when the LPI indication bit 7 is set in the network status register. Cleared on read."	RO RtoClr	0x00 0000	

tx_lpi

Register Information					
Description	"Transmit LPI transitions"				
Offset	0x278		**Type:**	RO	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:16	reserved_31_16	"Unused, read zero"	RO	0x0000	
15:0	count	"Count of TX LPI transmissions. A count of the number of times the enable LPI transmission bit 19 goes from low to high in the network control register. Cleared on read."	RO RtoClr	0x0000	

tx_lpi_time

Register Information			
Description	"Transmit LPI time"		
Offset	0x27C	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:24	reserved_31_24	"Unused, read zero"	RO	0x00
23:0	lpi_time	"Time in LPI. This register increments once every 16 pclk cycles when the enable LPI transmission bit 19 is set in the network control register. Cleared on read."	RO RtoClr	0x00 0000

designcfg_debug1

Register Information			
Description	"The GEM has many parameterisation options to configure the IP during compilation stage. This is achieved using Verilog define compiler directives in an include file called gem_defs.v. Refer to Section 3 for further details on how to configure the GEM in this way. This configuration is readable through APB addressable registers as follows :Design Configuration Register 1"		
Offset	0x280	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:28	axi_cache_value	"Takes the value of the `gem_axi_cache_value DEFINE"	RO	0x0
27:25	dma_bus_width	"Takes the value of bits 7:5 of the `gem_dma_bus_width DEFINE. So if the define is set to decimal 64 this will return binary 010."	RO	0x2
24	exclude_cbs	"Takes the value of the `gem_exclude_cbs DEFINE."	RO	0
23	irq_read_clear	"Takes the value of the `gem_irq_read_clear DEFINE"	RO	1
22	no_snapshot	"Takes the value of the `gem_no_snapshot DEFINE"	RO	0
21	no_stats	"Takes the value of the `gem_no_stats DEFINE"	RO	0
20	reserved_20	"Reserved, read as 1, ignored on write."	RO	1
19:15	user_in_width	"Takes the value of the `gem_user_in_width DEFINE or 1 if user_io not defined"	RO	0x10
14:10	user_out_width	"Takes the value of the `gem_user_out_width DEFINE or 1 if user_io not defined"	RO	0x10
9	user_io	"Takes the value of the `gem_user_io DEFINE"	RO	1
8	reserved_8	"Reserved, read as 1, ignored on write."	RO	1
7	reserved_7	"Reserved, read as 0, ignored on write."	RO	0
6	ext_fifo_interface	"Takes the value of the `gem_ext_fifo_interface DEFINE"	RO	0
5	reserved_5	"Reserved, read as 0, ignored on write."	RO	0
4	int_loopback	"Takes the value of the `gem_int_loopback DEFINE"	RO	1
3:2	reserved_3_2	"Reserved, read as 0, ignored on write."	RO	0x0
1	exclude_qbv	"Takes the value of the `gem_exclude_qbv DEFINE"	RO	0
0	no_pcs	"Takes the value of the `gem_no_pcs DEFINE"	RO	0

designcfg_debug2

Register Information			
Description	"Design Configuration Register 2"		
Offset	0x284	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31	spram	"Takes the value of the `gem_spram DEFINE"	RO	0
30	axi	"Takes the value of the `gem_axi DEFINE"	RO	1
29:26	tx_pbuf_addr	"Takes the value of the `gem_tx_pbuf_addr DEFINE"	RO	0xE
25:22	rx_pbuf_addr	"Takes the value of the `gem_rx_pbuf_addr DEFINE"	RO	0xB
21	tx_pkt_buffer	"Takes the value of the `gem_tx_pkt_buffer DEFINE"	RO	1
20	rx_pkt_buffer	"Takes the value of the `gem_rx_pkt_buffer DEFINE"	RO	1
19:16	hprot_value	"Takes the value of the `gem_hprot_value DEFINE"	RO	0x1
15:14	reserved_15_14	"Unused, read zero"	RO	0x0
13:0	jumbo_max_length	"Takes the value of the `gem_jumbo_max_length DEFINE"	RO	0x2800

designcfg_debug3

Register Information			
Description	"Design Configuration Register 3"		
Offset	0x288	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:30	reserved_31_30	"Unused, read zero"	RO	0x0
29:24	num_spec_add_filt ers	"Takes the value of the `num_spec_add_filters DEFINE"	RO	0x24
23:0	reserved_23_0	"Unused, read zero - reserved for `gem_rx_fifo_size DEFINE in internal FIFO mode"	RO	0x00 0000

designcfg_debug4

Register Information			
Description	"Design Configuration Register 4"		
Offset	0x28C	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:0	reserved_31_0	"Unused, read zero - reserved for `gem_tx_fifo_size DEFINE in internal FIFO mode"	RO	0x0000 0000

designcfg_debug5

Register Information			
Description	"Design Configuration Register 5"		
Offset	0x290	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:29	axi_prot_value	"Takes the value of the `gem_axi_prot_value DEFINE"	RO	0x2
28	tsu_clk	"Takes the value of the `gem_tsu_clk DEFINE"	RO	1
27:20	rx_buffer_length_def	"Takes the value of the `gem_rx_buffer_length_def DEFINE"	RO	0x02
19	tx_pbuf_size_def	"Takes the value of the `gem_tx_pbuf_size_def DEFINE"	RO	1
18:17	rx_pbuf_size_def	"Takes the value of the `gem_rx_pbuf_size_def DEFINE"	RO	0x3
16:15	endian_swap_def	"Takes the value of the `gem_endian_swap_def DEFINE"	RO	0x3
14:12	mdc_clock_div	"Takes the value of the `gem_mdc_clock_div DEFINE"	RO	0x2
11:10	dma_bus_width_def	"Takes the value of the `gem_dma_bus_width_def DEFINE"	RO	0x0
9	phy_ident	"Indicates whether the PHY_ID register is present"	RO	1
8	tsu	"Takes the value of the `gem_tsu DEFINE"	RO	1
7:4	tx_fifo_cnt_width	"Takes the value of the `gem_tx_fifo_cnt_width DEFINE"	RO	0x4
3:0	rx_fifo_cnt_width	"Takes the value of the `gem_rx_fifo_cnt_width DEFINE"	RO	0x5

designcfg_debug6

Register Information			
Description	"Design Configuration Register 6"		
Offset	0x294	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:28	reserved_31_28	"Reserved. Set to zero."	RO	0x0
27	pbuf_lso	"Takes the value of the `gem_pbuf_lso DEFINE"	RO	0
26	pbuf_rsc	"Takes the value of the `gem_pbuf_rsc DEFINE"	RO	1
25	pbuf_cutthru	"Takes the value of the `gem_pbuf_cutthru DEFINE"	RO	1
24	pfc_multi_quantum	"Takes the value of the `gem_pfc_multi_quantum DEFINE"	RO	1
23	dma_addr_width_is_64b	"Takes the value of the `gem_dma_addr_width_is_64b DEFINE"	RO	0
22	host_if_soft_select	"Takes the value of the `gem_host_if_soft_select DEFINE"	RO	1
21	tx_add_fifo_if	"Takes the value of the `gem_tx_add_fifo_if DEFINE"	RO	0
20	ext_tsu_timer	"Takes the value of the `gem_ext_tsu_timer DEFINE"	RO	1
19:16	tx_pbuf_queue_segment_size	"Takes the value of the `gem_tx_pbuf_queue_segment_size DEFINE"	RO	0x4
15	dma_priority_queue15	"Takes the value of the `dma_priority_queue15 DEFINE"	RO	1
14	dma_priority_queue14	"Takes the value of the `dma_priority_queue14 DEFINE"	RO	1
13	dma_priority_queue13	"Takes the value of the `dma_priority_queue13 DEFINE"	RO	1
12	dma_priority_queue12	"Takes the value of the `dma_priority_queue12 DEFINE"	RO	1
11	dma_priority_queue11	"Takes the value of the `dma_priority_queue11 DEFINE"	RO	1
10	dma_priority_queue10	"Takes the value of the `dma_priority_queue10 DEFINE"	RO	1
9	dma_priority_queue9	"Takes the value of the `dma_priority_queue9 DEFINE"	RO	1
8	dma_priority_queue8	"Takes the value of the `dma_priority_queue8 DEFINE"	RO	1
7	dma_priority_queue7	"Takes the value of the `dma_priority_queue7 DEFINE"	RO	1
6	dma_priority_queue6	"Takes the value of the `dma_priority_queue6 DEFINE"	RO	1
5	dma_priority_queue5	"Takes the value of the `dma_priority_queue5 DEFINE"	RO	1
4	dma_priority_queue4	"Takes the value of the `dma_priority_queue4 DEFINE"	RO	1
3	dma_priority_queue3	"Takes the value of the `dma_priority_queue3 DEFINE"	RO	1
2	dma_priority_queue2	"Takes the value of the `dma_priority_queue2 DEFINE"	RO	1
1	dma_priority_queue1	"Takes the value of the `dma_priority_queue1 DEFINE"	RO	1

0	reserved_0	"Reserved. Set to zero."	RO	0

designcfg_debug7

Register Information			
Description	"Design Configuration Register 7"		
Offset	0x298	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:28	tx_pbuf_num_segments_q7	"Takes the value of the `gem_tx_pbuf_num_segments_q7 DEFINE"	RO	0x0
27:24	tx_pbuf_num_segments_q6	"Takes the value of the `gem_tx_pbuf_num_segments_q6 DEFINE"	RO	0x0
23:20	tx_pbuf_num_segments_q5	"Takes the value of the `gem_tx_pbuf_num_segments_q5 DEFINE"	RO	0x0
19:16	tx_pbuf_num_segments_q4	"Takes the value of the `gem_tx_pbuf_num_segments_q4 DEFINE"	RO	0x0
15:12	tx_pbuf_num_segments_q3	"Takes the value of the `gem_tx_pbuf_num_segments_q3 DEFINE"	RO	0x0
11:8	tx_pbuf_num_segments_q2	"Takes the value of the `gem_tx_pbuf_num_segments_q2 DEFINE"	RO	0x0
7:4	tx_pbuf_num_segments_q1	"Takes the value of the `gem_tx_pbuf_num_segments_q1 DEFINE"	RO	0x0
3:0	tx_pbuf_num_segments_q0	"Takes the value of the `gem_tx_pbuf_num_segments_q0 DEFINE"	RO	0x0

designcfg_debug8

Register Information			
Description	"Design Configuration Register 8"		
Offset	0x29C	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:24	num_type1_screeners	"Takes the value of the `num_type1_screeners DEFINE"	RO	0x10
23:16	num_type2_screeners	"Takes the value of the `num_type2_screeners DEFINE"	RO	0x10
15:8	num_scr2_ethtype_regs	"Takes the value of the `num_scr2_ethtype_regs DEFINE"	RO	0x08
7:0	num_scr2_compare_regs	"Takes the value of the `num_scr2_compare_regs DEFINE"	RO	0x20

designcfg_debug9

Register Information			
Description	"Design Configuration Register 9"		
Offset	0x2A0	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:28	tx_pbuf_num_segments_q15	"Takes the value of the `gem_tx_pbuf_num_segments_q15 DEFINE"	RO	0x0
27:24	tx_pbuf_num_segments_q14	"Takes the value of the `gem_tx_pbuf_num_segments_q14 DEFINE"	RO	0x0
23:20	tx_pbuf_num_segments_q13	"Takes the value of the `gem_tx_pbuf_num_segments_q13 DEFINE"	RO	0x0
19:16	tx_pbuf_num_segments_q12	"Takes the value of the `gem_tx_pbuf_num_segments_q12 DEFINE"	RO	0x0
15:12	tx_pbuf_num_segments_q11	"Takes the value of the `gem_tx_pbuf_num_segments_q11 DEFINE"	RO	0x0
11:8	tx_pbuf_num_segments_q10	"Takes the value of the `gem_tx_pbuf_num_segments_q10 DEFINE"	RO	0x0
7:4	tx_pbuf_num_segments_q9	"Takes the value of the `gem_tx_pbuf_num_segments_q9 DEFINE"	RO	0x0
3:0	tx_pbuf_num_segments_q8	"Takes the value of the `gem_tx_pbuf_num_segments_q8 DEFINE"	RO	0x0

designcfg_debug10

Register Information			
Description	"Design Configuration Register 10"		
Offset	0x2A4	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:28	emac_bus_width	"Takes the value of the `gem_emac_bus_width DEFINE. 1 - The MAC has a datawidth of 32bits. 2 - The MAC has a datawidth of 64bits. 4 - The MAC has a datawidth of 128bits "	RO	0x2
27:24	tx_pbuf_data	"Takes the value of the `gem_tx_pbuf_data DEFINE. 1 - The TX DPRAM has a datawidth of 32bits. 2 - The TX DPRAM has a datawidth of 64bits. 4 - The TX DPRAM has a datawidth of 128bits "	RO	0x2
23:20	rx_pbuf_data	"Takes the value of the `gem_rx_pbuf_data DEFINE. 1 - The RX DPRAM has a datawidth of 32bits. 2 - The RX DPRAM has a datawidth of 64bits. 4 - RX The DPRAM has a datawidth of 128bits"	RO	0x2
19:16	axi_access_pipeline_bits	"Takes the value of the `gem_axi_access_pipeline_bits DEFINE"	RO	0x4
15:12	axi_tx_descr_rd_buff_bits	"Takes the value of the `gem_axi_tx_descr_rd_buff_bits DEFINE"	RO	0x4
11:8	axi_rx_descr_rd_buff_bits	"Takes the value of the `gem_axi_rx_descr_rd_buff_bits DEFINE"	RO	0x4
7:4	axi_tx_descr_wr_buff_bits	"Takes the value of the `gem_axi_tx_descr_wr_buff_bits DEFINE"	RO	0x4
3:0	axi_rx_descr_wr_buff_bits	"Takes the value of the `gem_axi_rx_descr_wr_buff_bits DEFINE"	RO	0x4

spec_add5_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x300	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add5_top

Register Information			
Description	"Specific Address Top"		
Offset	0x304	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add6_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x308	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add6_top

Register Information			
Description	"Specific Address Top"		
Offset	0x30C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add7_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x310	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add7_top

Register Information			
Description	"Specific Address Top"		
Offset	0x314	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add8_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x318	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add8_top

Register Information					
Description	"Specific Address Top"				
Offset	0x31C		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add9_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x320		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add9_top

Register Information					
Description	"Specific Address Top"				
Offset	0x324		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add10_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x328		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add10_top

Register Information					
Description	"Specific Address Top"				
Offset	0x32C		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."		RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."		RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."		RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."		RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."		RW	0x0000

spec_add11_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x330		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."		RW	0x0000 0000

spec_add11_top

Register Information					
Description	"Specific Address Top"				
Offset	0x334		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add12_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x338		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add12_top

Register Information					
Description	"Specific Address Top"				
Offset	0x33C		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add13_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x340		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add13_top

Register Information			
Description	"Specific Address Top"		
Offset	0x344	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add14_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x348	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add14_top

Register Information			
Description	"Specific Address Top"		
Offset	0x34C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add15_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x350	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add15_top

Register Information			
Description	"Specific Address Top"		
Offset	0x354	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add16_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x358	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add16_top

Register Information				
Description	"Specific Address Top"			
Offset	0x35C	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add17_bottom

Register Information				
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "			
Offset	0x360	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add17_top

Register Information			
Description	"Specific Address Top"		
Offset	0x364	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add18_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x368	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add18_top

Register Information			
Description	"Specific Address Top"		
Offset	0x36C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add19_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x370	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add19_top

Register Information					
Description		"Specific Address Top"			
Offset		0x374	**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add20_bottom

Register Information					
Description		"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "			
Offset		0x378	**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add20_top

Register Information					
Description	"Specific Address Top"				
Offset	0x37C		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add21_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x380		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add21_top

Register Information				
Description	"Specific Address Top"			
Offset	0x384	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add22_bottom

Register Information				
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "			
Offset	0x388	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add22_top

Register Information			
Description	"Specific Address Top"		
Offset	0x38C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add23_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x390	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add23_top

Register Information				
Description	"Specific Address Top"			
Offset	0x394	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add24_bottom

Register Information				
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "			
Offset	0x398	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add24_top

Register Information					
Description	"Specific Address Top"				
Offset	0x39C		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add25_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x3A0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add25_top

Register Information			
Description	"Specific Address Top"		
Offset	0x3A4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add26_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x3A8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add26_top

Register Information					
Description	"Specific Address Top"				
Offset	0x3AC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add27_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x3B0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add27_top

Register Information					
Description	"Specific Address Top"				
Offset	0x3B4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add28_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x3B8		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add28_top

Register Information			
Description	"Specific Address Top"		
Offset	0x3BC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add29_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x3C0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add29_top

Register Information			
Description	"Specific Address Top"		
Offset	0x3C4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add30_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x3C8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add30_top

Register Information					
Description	"Specific Address Top"				
Offset	0x3CC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add31_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x3D0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add31_top

Register Information			
Description	"Specific Address Top"		
Offset	0x3D4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add32_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x3D8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add32_top

Register Information					
Description	"Specific Address Top"				
Offset	0x3DC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add33_bottom

Register Information				
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "			
Offset	0x3E0	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add33_top

Register Information			
Description	"Specific Address Top"		
Offset	0x3E4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add34_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x3E8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add34_top

Register Information			
Description	"Specific Address Top"		
Offset	0x3EC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

spec_add35_bottom

Register Information			
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "		
Offset	0x3F0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000

spec_add35_top

Register Information					
Description	"Specific Address Top"				
Offset	0x3F4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0	
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00	
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00	
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0	
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000	

spec_add36_bottom

Register Information					
Description	"The addresses stored in the specific address registers are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. "				
Offset	0x3F8		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:0	address	"Least significant 32 bits of the destination address, that is bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received."	RW	0x0000 0000	

spec_add36_top

Register Information				
Description	"Specific Address Top"			
Offset	0x3FC	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29:24	filter_byte_mask	"When high, the associated byte of the specific address will not be compared. Bit 24 controls whether the first byte received should be compared. Bit 29 controls whether the last byte received should be compared."	RW	0x00
23:17	reserved_23_17	"Reserved, read as 0, ignored on write."	RO	0x00
16	filter_type	"This control bit selects whether this filter should be comparing the MAC source address or the MAC destination address of the received Ethernet frame. When set to zero, the filter is a destination address filter. When set to one, the filter is a source address filter."	RW	0
15:0	address	"Specific address 1. The most significant bits of the destination/source address that is to be compared, that is bits 47:32."	RW	0x0000

int_q1_status

Register Information				
Description	"Priority Queue Interrupt Status Register"			
Offset	0x400	**Type:**	RO	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q2_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x404	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q3_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x408	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q4_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x40C	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q5_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x410	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q6_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x414	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q7_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x418	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q8_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x41C	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q9_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x420	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q10_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x424	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q11_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x428	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q12_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x42C	**Type:**	RO

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q13_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x430	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q14_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x434	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q15_status

Register Information			
Description	"Priority Queue Interrupt Status Register"		
Offset	0x438	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok	"bresp/hresp not OK "	RO	0
10	receive_overrun	"Receive overrun "	RO	0
9:8	reserved_9_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete	"Transmit complete "	RO	0
6	amba_error	"Transmit frame corruption due to AMBA (AHB/AXI) error set if an error occurs whilst midway through reading transmit frame from the external memory, including HRESP errors(AHB), RRESP and BRESP errors (AXI) and buffers exhausted mid frame "	RO	0
5	retry_limit_exceeded_or_late_collision	"Retry limit exceeded or late collision "	RO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_bit_read	"RX used bit read "	RO	0
1	receive_complete	"Receive complete "	RO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

transmit_q1_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x440	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q2_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x444	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q3_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x448	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q4_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x44C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q5_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x450	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q6_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x454	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q7_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x458	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q8_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x45C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q9_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x460	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q10_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x464	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q11_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x468	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q12_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x46C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q13_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x470	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q14_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x474	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

transmit_q15_ptr

Register Information			
Description	"This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The transmit buffer queue base address register must be initialized before transmit is started through bit 9 of the network control register. Once transmission has started, any write to the transmit buffer queue base address register is illegal and therefore ignored. Note that due to clock boundary synchronization, it takes a maximum of four pclk cycles from the writing of the transmit start bit before the transmitter is active. Writing to the transmit buffer queue base address register during this time may produce unpredictable results. Reading this register returns the location of the descriptor currently being accessed. Because the DMA can store data for multiple frames at once, this may not necessarily be pointing to the current frame being transmitted. In terms of AMBA AHB/AXI operation, the transmit descriptors are written to memory using a single 32bit AHB access. When the datapath is configured as 64bit or 128bit, the transmit descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is read from memory using a single AHB/AXI access. For 32bit datapaths, the descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual 32-bit non sequential accesses."		
Offset	0x478	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_tx_q_ptr	"Transmit buffer queue base address - written with the address of the start of the transmit queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_tx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while transmit is not enabled."	RW	0

receive_q1_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x480	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q2_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x484	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q3_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x488	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q4_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x48C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q5_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x490	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q6_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x494	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q7_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x498	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

dma_rxbuf_size_q1

Register Information			
Description	"Receive Buffer Queue Size"		
Offset	0x4A0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02

dma_rxbuf_size_q2

Register Information				
Description	"Receive Buffer Queue Size"			
Offset	0x4A4	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02

dma_rxbuf_size_q3

Register Information				
Description	"Receive Buffer Queue Size"			
Offset	0x4A8	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02

dma_rxbuf_size_q4

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x4AC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000	
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02	

dma_rxbuf_size_q5

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x4B0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000	
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02	

dma_rxbuf_size_q6

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x4B4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."		RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."		RW	0x02

dma_rxbuf_size_q7

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x4B8		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**		**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."		RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."		RW	0x02

cbs_control

Register Information			
Description	"The IdleSlope value is defined as the rate of change of credit when a packet is waiting to be sent. This must not exceed the portTransmitRate which is dependent on the speed of operation, eg, portTranmsitRate: 1Gb/s = 32'h07735940 (125 Mbytes/s), 100Mb/sec = 32'h017D7840 (25 Mnibbles/s), 10Mb/sec = 32'h002625A0 (2.5 Mnibbles/s). If 50% of bandwidth was to be allocated to a particular queue in 1Gb/sec mode then the IdleSlope value for that queue would be calculated as 32'h07735940/2. Note: Credit-Based Shaping should be disabled prior to updating the IdleSlope values. As another example, for a 1722 audio packet with a payload of 6 samples per channel, the packet size would be: 7 (preamble) + 1 (SFD) + 50 (packet header) + 6x4x2(payload) + 4 (CRC) = 110 bytes. For a rate of 8000 packets per second, the desired rate would 110 x 8000 bytes per second, so the programmed idleSlope value would be 880000 for a 1Gb/s connection, or 1760000 for a 100Mb/s or 10Mbs connection. See Figure 6.3 in the IEEE 1722 standard. In practice, the actual transmission rate will be vary slightly from that calculated. In this case, the idleSlope value should be recalibrated based on the variance between the measured and expected rate, and in this case very accurate transmission rates can be achieved."		
Offset	0x4BC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	reserved_31_2	"Reserved, read as 0, ignored on write."	RO	0x0000 0000
1	cbs_enable_queue_b	"Enable Credit-Based shaping on the 2nd highest priority queue (queue B). Write 1 to enable"	RW	0
0	cbs_enable_queue_a	"Enable Credit-Based Shaping on the highest priority queue (queue A). Write 1 to enable"	RW	0

cbs_idleslope_q_a

Register Information			
Description	"Queue A is the highest priority queue. This is the highest indexed active queue, e.g. For a system with Q0 to Q7, this would be Q7 if all queues were active."		
Offset	0x4C0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	idleslope_a	"IdleSlope value for queue A in bytes/sec for gigabit operation and nibbles/sec for 10/100 operation"	RW	0x0000 0000

cbs_idleslope_q_b

Register Information			
Description	"Queue B is the 2nd highest priority queue. This is the second highest indexed active queue, e.g. For a system with Q0 to Q7, this would be Q6 if all queues were active."		
Offset	0x4C4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	idleslope_b	"IdleSlope value for queue B in bytes/sec for gigabit operation and nibbles/sec for 10/100 operation"	RW	0x0000 0000

upper_tx_q_base_addr

Register Information			
Description	"Upper 32 bits of transmit buffer descriptor queue base address."		
Offset	0x4C8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	upper_tx_q_base_addr	"Upper 32 bits of transmit buffer descriptor queue base address. Used when 64 bit addressing is enabled. (In releases earlier to 1p06f2 this register also affected the receive descriptor queue.)"	RW	0x0000 0000

tx_bd_control

Register Information			
Description	"TX BD control register"		
Offset	0x4CC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:6	reserved_31_6	"Reserved, read as 0, ignored on write."	RO	0x000 0000
5:4	tx_bd_ts_mode	"TX Descriptor Timestamp Insertion mode, 00: TS insertion disable, 01: TS inserted for PTP Event Frames only, 10: TS inserted for All PTP Frames only, 11: TS insertion for All Frames "	RW	0x0
3:0	reserved_3_0	"Reserved, read as 0, ignored on write."	RO	0x0

rx_bd_control

Register Information			
Description	"RX BD control register"		
Offset	0x4D0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:6	reserved_31_6	"Reserved, read as 0, ignored on write."	RO	0x000 0000
5:4	rx_bd_ts_mode	"RX Descriptor Timestamp Insertion mode, 00: TS insertion disable, 01: TS inserted for PTP Event Frames only, 10: TS inserted for All PTP Frames only, 11: TS insertion for All Frames "	RW	0x0
3:0	reserved_3_0	"Reserved, read as 0, ignored on write."	RO	0x0

upper_rx_q_base_addr

Register Information			
Description	"Upper 32 bits of receive buffer descriptor queue base address."		
Offset	0x4D4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:0	upper_rx_q_base_addr	"Upper 32 bits of receive buffer descriptor queue base address. Used when 64 bit addressing is enabled."	RW	0x0000 0000

screening_type_1_register_0

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x500	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_1

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x504	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_2

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x508	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_3

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x50C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_4

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x510	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_5

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x514	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_6

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x518	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_7

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x51C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_8

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x520	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_9

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x524	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_10

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x528	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_11

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x52C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_12

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x530	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_13

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x534	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_14

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x538	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_1_register_15

Register Information			
Description	"Screening type 1 registers are used to allocate up to 16 priority queues to received frames based on certain IP or UDP fields of incoming frames. Firstly, when DS/TC match enable is set (bit 28), the DS (Differentiated Services) field of the received IPv4 header or TCfield (traffic class) of IPv6 headers are matched against bits 11:4. Secondly, when UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12. Both UDP and DS/TC matching can be enabled simultaneously or individually. If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame. The required number of Type 1 screening registers is configured in the gem defines file. Up to 16 type 1 screening registers have been allocated APB address space between 0x500 and 0x53C. The bit mappings for these registers will be as follows:"		
Offset	0x53C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	reserved_31_30	"Reserved, read as 0, ignored on write."	RO	0x0
29	udp_port_match_enable	"UDP port match enable"	RW	0
28	dstc_enable	"DS/TC Enable"	RW	0
27:12	udp_port_match	"UDP Port Match"	RW	0x0000
11:4	dstc_match	"DS/TC Match"	RW	0x00
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_0

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x540	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_1

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x544	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_2

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x548	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_3

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x54C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_4

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x550	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_5

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x554	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_6

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x558	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_7

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x55C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_8

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x560	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_9

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x564	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_10

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x568	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_11

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x56C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_12

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x570	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_13

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x574	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_14

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x578	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

screening_type_2_register_15

Register Information			
Description	"Screener Type 2 match registers allow a screen to be configured that is the combination of all or any of the following comparisons 1) An enabled VLAN Priority. A VLAN Priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against 3 bits within the screener type 2 register itself. 2) An enabled EtherType. 3) An enabled Field Compare A. 4) An enabled Field Compare B. 5) An enabled Field Compare C. The required number of Type 2 screening registers is configured in the gem defines file. Up to 16 type 2 screening registers have been allocated APB address space between 0x540 and 0x57C. The bit mappings for these registers is as follows:"		
Offset	0x57C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31	"Reserved, read as 0, ignored on write."	RO	0
30	compare_c_enable	"Compare C Enable"	RW	0
29:25	compare_c	"Compare C - Index to screener type 2 Compare register"	RW	0x00
24	compare_b_enable	"Compare B Enable"	RW	0
23:19	compare_b	"Compare B - Index to screener type 2 Compare register"	RW	0x00
18	compare_a_enable	"Compare A Enable"	RW	0
17:13	compare_a	"Compare A - Index to screener type 2 Compare register "	RW	0x00
12	ethertype_enable	"EtherType Enable"	RW	0
11:9	index	"Index to screener type 2 EtherType register"	RW	0x0
8	vlan_enable	"VLAN Enable"	RW	0
7	reserved_7	"Reserved and implemented as RW"	RW	0
6:4	vlan_priority	"VLAN Priority"	RW	0x0
3:0	queue_number	"Queue Number (0 to 15)"	RW	0x0

tx_sched_ctrl

Register Information			
Description	"This register controls the transmit scheduling algorithm the user can select for each active transmit queue. By default all queues are initialized to fixed priority, with the top indexed queue having overall priority"		
Offset	0x580	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:30	tx_sched_q16	"Queue 15 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
29:28	tx_sched_q15	"Queue 14 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
27:26	tx_sched_q14	"Queue 13 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
25:24	tx_sched_q13	"Queue 12 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
23:22	tx_sched_q12	"Queue 11 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
21:20	tx_sched_q11	"Queue 10 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
19:18	tx_sched_q10	"Queue 9 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
17:16	tx_sched_q9	"Queue 8 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0

15:14	tx_sched_q8	"Queue 7 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
13:12	tx_sched_q7	"Queue 6 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
11:10	tx_sched_q6	"Queue 5 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
9:8	tx_sched_q5	"Queue 4 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
7:6	tx_sched_q4	"Queue 3 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
5:4	tx_sched_q3	"Queue 2 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
3:2	tx_sched_q2	"Queue 1 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0
1:0	tx_sched_q1	"Queue 0 selection. 00 : Fixed Priority 01 : CBS Enabled only valid for top two enabled queues and if CBS capability selected. 10 : DWRR Enabled 11 : ETS Enabled"	RW	0x0

bw_rate_limit_q0to3

Register Information			
Description	"This register holds the DWRR weighting value or the ETS bandwidth percentage value used by the transmit scheduler for queues 0 to 0."		
Offset	0x590	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:24	dwrr_ets_weight_q3	"DWRR Weighting / ETS Bandwidth Allocation for queue 3"	RW	0x00
23:16	dwrr_ets_weight_q2	"DWRR Weighting / ETS Bandwidth Allocation for queue 2"	RW	0x00
15:8	dwrr_ets_weight_q1	"DWRR Weighting / ETS Bandwidth Allocation for queue 1"	RW	0x00
7:0	dwrr_ets_weight_q0	"DWRR Weighting / ETS Bandwidth Allocation for queue 0"	RW	0x00

bw_rate_limit_q4to7

Register Information			
Description	"This register holds the DWRR weighting value or the ETS bandwidth percentage value used by the transmit scheduler for queues 4 to 0."		
Offset	0x594	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:24	dwrr_ets_weight_q7	"DWRR Weighting / ETS Bandwidth Allocation for queue 7"	RW	0x00
23:16	dwrr_ets_weight_q6	"DWRR Weighting / ETS Bandwidth Allocation for queue 6"	RW	0x00
15:8	dwrr_ets_weight_q5	"DWRR Weighting / ETS Bandwidth Allocation for queue 5"	RW	0x00
7:0	dwrr_ets_weight_q4	"DWRR Weighting / ETS Bandwidth Allocation for queue 4"	RW	0x00

bw_rate_limit_q8to11

Register Information			
Description	"This register holds the DWRR weighting value or the ETS bandwidth percentage value used by the transmit scheduler for queues 8 to 0."		
Offset	0x598	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:24	dwrr_ets_weight_q11	"DWRR Weighting / ETS Bandwidth Allocation for queue 11"	RW	0x00
23:16	dwrr_ets_weight_q10	"DWRR Weighting / ETS Bandwidth Allocation for queue 10"	RW	0x00
15:8	dwrr_ets_weight_q9	"DWRR Weighting / ETS Bandwidth Allocation for queue 9"	RW	0x00
7:0	dwrr_ets_weight_q8	"DWRR Weighting / ETS Bandwidth Allocation for queue 8"	RW	0x00

bw_rate_limit_q12to15

Register Information			
Description	"This register holds the DWRR weighting value or the ETS bandwidth percentage value used by the transmit scheduler for queues 12 to 0."		
Offset	0x59C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:24	dwrr_ets_weight_q15	"DWRR Weighting / ETS Bandwidth Allocation for queue 15"	RW	0x00
23:16	dwrr_ets_weight_q14	"DWRR Weighting / ETS Bandwidth Allocation for queue 14"	RW	0x00
15:8	dwrr_ets_weight_q13	"DWRR Weighting / ETS Bandwidth Allocation for queue 13"	RW	0x00
7:0	dwrr_ets_weight_q12	"DWRR Weighting / ETS Bandwidth Allocation for queue 12"	RW	0x00

tx_q_seg_alloc_q0to7

Register Information			
Description	"This register allows the user to distribute the Transmit SRAM used by the DMA across the priority queues, for queues 0 to 7. The SRAM itself is split into a number of evenly sized segments (this is defined in the verilog configuration defs file - for the configuration used to generate this register description, the total number of segments was set to '16'). Those segments can then be freely distributed across the active queues, in powers of 2. I.e. a value of 0 would mean 1 segment has been allocated to the queue. A value of 1 would mean 2 segments, a value of 2 means 4 segments and so on. The reset values of these registers are defined in the configuration defs file."		
Offset	0x5A0	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31_31	"Reserved, read as 0, ignored on write."	RO	0
30:28	segment_alloc_q7	"Number of segments allocated to q7. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
27	reserved_27_27	"Reserved, read as 0, ignored on write."	RO	0
26:24	segment_alloc_q6	"Number of segments allocated to q6. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
23	reserved_23_23	"Reserved, read as 0, ignored on write."	RO	0
22:20	segment_alloc_q5	"Number of segments allocated to q5. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
19	reserved_19_19	"Reserved, read as 0, ignored on write."	RO	0
18:16	segment_alloc_q4	"Number of segments allocated to q4. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
15	reserved_15_15	"Reserved, read as 0, ignored on write."	RO	0
14:12	segment_alloc_q3	"Number of segments allocated to q3. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
11	reserved_11_11	"Reserved, read as 0, ignored on write."	RO	0
10:8	segment_alloc_q2	"Number of segments allocated to q2. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
7	reserved_7_7	"Reserved, read as 0, ignored on write."	RO	0
6:4	segment_alloc_q1	"Number of segments allocated to q1. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
3	reserved_3_3	"Reserved, read as 0, ignored on write."	RO	0
2:0	segment_alloc_q0	"Number of segments allocated to q0. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0

tx_q_seg_alloc_q8to15

Register Information			
Description	"This register allows the user to distribute the Transmit SRAM used by the DMA across the priority queues, for queues 8 to 15. The SRAM itself is split into a number of evenly sized segments (this is defined in the verilog configuration defs file - for the configuration used to generate this register description, the total number of segments was set to '16'). Those segments can then be freely distributed across the active queues, in powers of 2. I.e. a value of 0 would mean 1 segment has been allocated to the queue. A value of 1 would mean 2 segments, a value of 2 means 4 segments and so on. The reset values of these registers are defined in the configuration defs file."		
Offset	0x5A4	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31	reserved_31_31	"Reserved, read as 0, ignored on write."	RO	0
30:28	segment_alloc_q15	"Number of segments allocated to q15. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
27	reserved_27_27	"Reserved, read as 0, ignored on write."	RO	0
26:24	segment_alloc_q14	"Number of segments allocated to q14. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
23	reserved_23_23	"Reserved, read as 0, ignored on write."	RO	0
22:20	segment_alloc_q13	"Number of segments allocated to q13. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
19	reserved_19_19	"Reserved, read as 0, ignored on write."	RO	0
18:16	segment_alloc_q12	"Number of segments allocated to q12. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
15	reserved_15_15	"Reserved, read as 0, ignored on write."	RO	0
14:12	segment_alloc_q11	"Number of segments allocated to q11. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
11	reserved_11_11	"Reserved, read as 0, ignored on write."	RO	0
10:8	segment_alloc_q10	"Number of segments allocated to q10. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
7	reserved_7_7	"Reserved, read as 0, ignored on write."	RO	0
6:4	segment_alloc_q9	"Number of segments allocated to q9. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0
3	reserved_3_3	"Reserved, read as 0, ignored on write."	RO	0
2:0	segment_alloc_q8	"Number of segments allocated to q8. This should be entered as a log 2, for example entering a value of 2 would grant 4 segments. A maximum of 32 segments can be granted."	RW	0x0

receive_q8_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5C0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q9_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5C4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q10_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5C8	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q11_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5CC	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q12_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5D0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q13_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5D4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q14_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5D8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

receive_q15_ptr

Register Information			
Description	"This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the network control register. Once reception is enabled, any write to the receive buffer queue base address register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits. In terms of AMBA (AHB/AXI) operation, the receive descriptors are read from memory using a single 32bit AHB/AXI access. When the datapath is configured at 64bit or 128bit, the receive descriptors should be aligned at 64-bit boundaries and each pair of 32-bit descriptors is written to using a single 64bit AHB/AXI access. For 32bit datapaths, the receive descriptors should be aligned at 32-bit boundaries and are written to using two individual non sequential 32-bit accesses. "		
Offset	0x5DC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:2	dma_rx_q_ptr	"Receive buffer queue base address - written with the address of the start of the receive queue."	RW	0x0000 0000
1	reserved_1_1	"Reserved, read as 0, ignored on write."	RO	0
0	dma_rx_dis_q	"Disable queue if set to 1. This can be used to reduce the number of active queues and should only be changed while receive is not enabled."	RW	0

dma_rxbuf_size_q8

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x5E0		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000	
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02	

dma_rxbuf_size_q9

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x5E4		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Access**	**Reset**	
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000	
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02	

dma_rxbuf_size_q10

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x5E8		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000	
7:0	dma_rx_q_buf_siz e	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02	

dma_rxbuf_size_q11

Register Information					
Description	"Receive Buffer Queue Size"				
Offset	0x5EC		**Type:**	RW	
Bitfield Details					
Bits	**Name**	**Description**	**Acces s**	**Reset**	
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000	
7:0	dma_rx_q_buf_siz e	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02	

dma_rxbuf_size_q12

Register Information				
Description	"Receive Buffer Queue Size"			
Offset	0x5F0	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02

dma_rxbuf_size_q13

Register Information				
Description	"Receive Buffer Queue Size"			
Offset	0x5F4	**Type:**	RW	
Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02

dma_rxbuf_size_q14

Register Information			
Description	"Receive Buffer Queue Size"		
Offset	0x5F8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02

dma_rxbuf_size_q15

Register Information			
Description	"Receive Buffer Queue Size"		
Offset	0x5FC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:8	reserved_31_8	"Reserved, read as 0, ignored on write."	RO	0x00 0000
7:0	dma_rx_q_buf_size	"DMA receive buffer size in system memory. The value defined by these bits determines the size of buffer to use in main system memory when writing received data. The value is defined in multiples of 64 bytes. 0x01 corresponds to buffers of 64 bytes. 0x02 corresponds to 128 bytes etc. For example: 0x02: 128 byte 0x18: 1536 byte (1*max length frame/buffer) 0xA0: 10240 byte (1*10K jumbo frame/buffer) Note that this value should never be written as zero. Note. The reset value of this field is equal to the gem_rx_buffer_length_def define, which is user configurable."	RW	0x02

int_q1_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x600	**Type:**	RW

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q2_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x604	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q3_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x608	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q4_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x60C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q5_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x610	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q6_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x614	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q7_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x618	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q1_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x620	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q2_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x624	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q3_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x628	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q4_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x62C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q5_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x630	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q6_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x634	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q7_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x638	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q1_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x640	**Type:**	RO

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q2_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x644	**Type:**	RO

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q3_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x648	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q4_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x64C	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q5_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x650	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q6_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x654	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q7_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x658	**Type:**	RO

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q8_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x660	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q9_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x664	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q10_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x668	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q11_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x66C	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q12_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x670	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q13_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x674	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q14_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x678	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q15_enable

Register Information			
Description	"At reset all interrupts are disabled. Writing a one to the relevant bit location enables the required interrupt. This register is write only and when read will return zero."		
Offset	0x67C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	enable_resp_not_ok_interrupt	"Enable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	enable_transmit_complete_interrupt	"Enable Transmit complete interrupt"	WO	0
6	enable_transmit_frame_corruption_due_to_amba_error_interrupt	"Enable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	enable_retry_limit_exceeded_or_late_collision_interrupt	"Enable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	enable_rx_used_bit_read_interrupt	"Enable RX used bit read interrupt"	WO	0
1	enable_receive_complete_interrupt	"Enable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q8_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x680	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q9_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x684	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q10_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x688	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q11_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x68C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q12_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x690	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q13_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x694	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q14_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x698	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q15_disable

Register Information			
Description	"Writing a 1 to the relevant bit location disables that particular interrupt. This register is write only and when read will return zero."		
Offset	0x69C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	disable_resp_not_ok_interrupt	"Disable bresp/hresp not OK interrupt"	WO	0
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	disable_transmit_complete_interrupt	"Disable Transmit complete interrupt"	WO	0
6	disable_transmit_frame_corruption_due_to_amba_error_interrupt	"Disable Transmit frame corruption due to AMBA (AHB/AXI) error interrupt"	WO	0
5	disable_retry_limit_exceeded_or_late_collision_interrupt	"Disable Retry limit exceeded or late collision interrupt"	WO	0
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	disable_rx_used_bit_read_interrupt	"Disable RX used bit read interrupt"	WO	0
1	disable_receive_complete_interrupt	"Disable Receive complete interrupt"	WO	0
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q8_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6A0	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q9_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6A4	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q10_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6A8	**Type:**	RO

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q11_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6AC	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q12_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6B0	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q13_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6B4	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q14_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6B8	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

int_q15_mask

Register Information			
Description	"The interrupt mask register is a read only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the interrupt enable register or set individually by writing to the interrupt disable register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the interrupt mask register. For test purposes there is a write-only function to this register that allows the bits in the interrupt status register to be set or cleared, regardless of the state of the mask register."		
Offset	0x6BC	**Type:**	RO

Bitfield Details

Bits	Name	Description	Access	Reset
31:12	reserved_31_12	"Reserved, read as 0, ignored on write."	RO	0x0 0000
11	resp_not_ok_interrupt_mask	"bresp/hresp not OK interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
10:8	reserved_10_8	"Reserved, read as 0, ignored on write."	RO	0x0
7	transmit_complete_interrupt_mask	"transmit complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
6	amba_error_interrupt_mask	"A read of this register returns the value of the AMBA (AHB/AXI) error interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
5	retry_limit_exceeded_or_late_collision_interrupt_mask	"retry limit exceeded or late collision interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
4:3	reserved_4_3	"Reserved, read as 0, ignored on write."	RO	0x0
2	rx_used_interrupt_mask	"A read of this register returns the value of the RX Used interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
1	receive_complete_interrupt_mask	"receive complete interrupt mask. 0: Interrupt is enabled. 1: Interrupt is disabled. A write to this register directly affects the state of the corresponding bit in the interrupt status register, causing an interrupt to be generated if a 1 is written."	RO	1
0	reserved_0	"Reserved, read as 0, ignored on write."	RO	0

screening_type_2_ethertype_reg_0

Register Information			
Description	"Ethertype Register"		
Offset	0x6E0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

screening_type_2_ethertype_reg_1

Register Information			
Description	"Ethertype Register"		
Offset	0x6E4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

screening_type_2_ethertype_reg_2

Register Information			
Description	"Ethertype Register"		
Offset	0x6E8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

screening_type_2_ethertype_reg_3

Register Information			
Description	"Ethertype Register"		
Offset	0x6EC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

screening_type_2_ethertype_reg_4

Register Information			
Description	"Ethertype Register"		
Offset	0x6F0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

screening_type_2_ethertype_reg_5

Register Information			
Description	"Ethertype Register"		
Offset	0x6F4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

screening_type_2_ethertype_reg_6

Register Information			
Description	"Ethertype Register"		
Offset	0x6F8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

screening_type_2_ethertype_reg_7

Register Information			
Description	"Ethertype Register"		
Offset	0x6FC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	reserved_31_16	"Reserved, read as 0, ignored on write "	RO	0x0000
15:0	compare_value	"Ethertype compare value"	RW	0x0000

type2_compare_0_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x700	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_0_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x704	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_1_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x708	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_1_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x70C	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_2_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x710	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_2_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x714	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_3_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x718	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_3_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x71C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_4_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x720	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_4_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x724	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_5_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x728	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_5_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x72C	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_6_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x730	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_6_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x734	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_7_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x738	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_7_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x73C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_8_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x740	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_8_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x744	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_9_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x748	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_9_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x74C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_10_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x750	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_10_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x754	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_11_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x758	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_11_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x75C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_12_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x760	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_12_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x764	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_13_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x768	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_13_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x76C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_14_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x770	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_14_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x774	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_15_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x778	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_15_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x77C	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_16_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x780	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_16_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x784	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_17_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x788	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_17_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x78C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_18_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x790	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_18_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x794	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_19_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x798	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_19_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x79C	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_20_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7A0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_20_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7A4	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_21_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7A8	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_21_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7AC	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_22_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7B0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_22_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7B4	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_23_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7B8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_23_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7BC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_24_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7C0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_24_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7C4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_25_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7C8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_25_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7CC	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_26_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7D0	**Type:**	RW

Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_26_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7D4	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Acces s**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_27_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7D8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_27_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7DC	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_28_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7E0	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_28_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7E4	**Type:**	RW

Bitfield Details

Bits	Name	Description	Access	Reset
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_29_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7E8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_29_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7EC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_30_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7F0	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_30_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7F4	**Type:**	RW

Bitfield Details

Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

type2_compare_31_word_0

Register Information			
Description	"Compare A, B and C fields of the screener type 2 match register are pointers to a pool of up to 32 compare registers. If enabled the compare is true if the data at the OFFSET into the frame, ANDed with the MASK Value if the mask is enabled, is equal to the COMPARE Value. Either a 16 bit comparison or a 32 bit comparison is done. This selection is made via the associated compare word1 register bit 9. If a 16 bit comparison is selected, then a 16 bit mask is also available to the user to select which bits should be compared. If the 32 bit compare option is selected, then no mask is available. The byte at the OFFSET number of bytes from the index start is compared thru bits 7:0 of the configured VALUE. The byte at the OFFSET number of bytes + 1 from the index start is compared thru bits 15:8 of the configured VALUE and so on. The OFFSET can be configured to be from 0 to 127 bytes from either the start of the frame, the byte following the etherType field (last EtherType in the header if the frame is VLAN tagged), the byte following the IP header (IPv4 or IPv6) or from the byte following the start of the TCP/UDP header. The required number of Type 2 screening registers up to a maximum of 32 is configurable in the gem defines file and have been allocated APB address space between 0x700 and 0x7fc. Note. when using RX Partial Store and Forward mode and priority queues, the frame offset must be less than the Partial Store and Forward watermark. If the offset is higher than the watermark value it's not possible to identify the priority queue before the frame is sent to the AMBA interface, and an incorrect priority queue may be used. The bit mapping for these registers is as follows:"		
Offset	0x7F8	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:16	compare_value	"2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+2 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+3. If bit 9 of the associated compare_word1 register is clear, then the byte stored in bits [23:16] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [31:24] is compared against the byte in the received frame from the selected offset+1. "	RW	0x0000
15:0	mask_value	"These bits can be either a 2 byte mask field or an additional 2 byte Compare Value. If bit 9 of the associated compare_word1 register is set, then the byte stored in bits [7:0] is compared against the byte in the received frame from the selected offset+0 and the byte stored in bits [15:8] is compared against the byte in the received frame from the selected offset+1. If bit 9 of the associated compare_word1 register is clear, these bits become a direct 2-byte mask for the 2-byte compare register in bits [31:16]."	RW	0x0000

type2_compare_31_word_1

Register Information			
Description	"Type2 Compare Word 1"		
Offset	0x7FC	**Type:**	RW

Bitfield Details				
Bits	**Name**	**Description**	**Access**	**Reset**
31:10	reserved_31_10	"Reserved, read as 0, ignored on write."	RO	0x00 0000
9	disable_mask	"This bit is used to control whether the compare register word_0 contains a 4-byte compare value, or a 2-byte compare value with a 2-byte mask value. 1 - 4-byte compare value 0 - 2-byte compare, 2-byte mask "	RW	0
8:7	compare_offset	"compare byte offset. 00 : Offset from beginning of the frame. 01 : Offset from byte after Ether Type. 10 : Offset from byte following end of IP header. 11 : Offset from byte following end of TCP/UDP header"	RW	0x0
6:0	offset_value	"Offset value in bytes"	RW	0x00

Programming Examples

Initialization

Configuration

Initialization of the GEM_GXL configuration must be done while the transmit and receive circuits are disabled. See the description of the network control register and network configuration register earlier in this document.

To change loop back mode, the following sequence of operations **must** be followed:

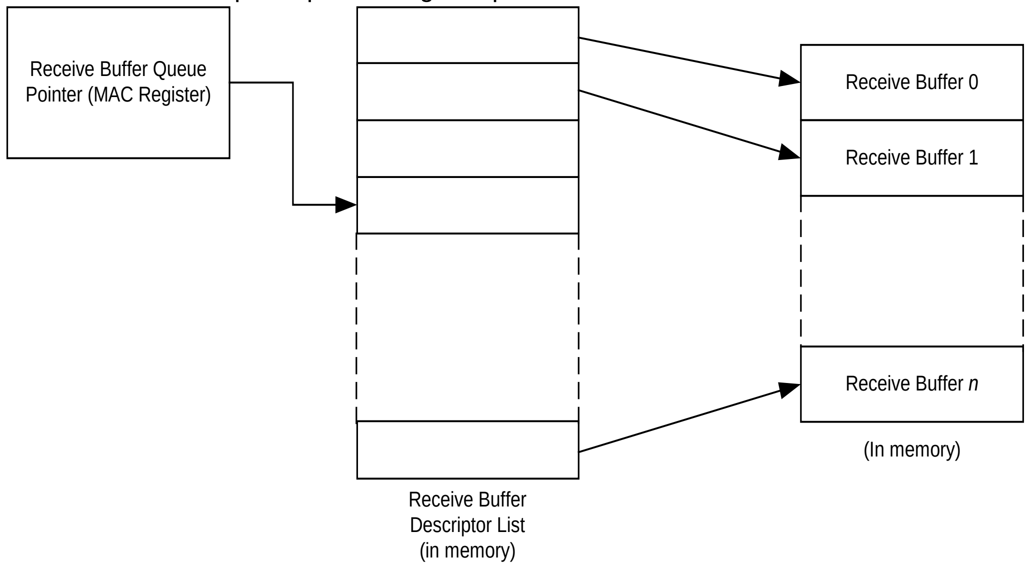
1. Write to network control register to disable transmit and receive circuits.
2. Write to network control register to change loop back mode.
3. Write to network control register to re-enable transmit or receive circuits.

Note: *These writes to network control register cannot be combined in any way.*

Receive Buffer List

Receive data is written to areas of data (i.e. buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (receive buffer queue) is a sequence of descriptor entries as defined in the table Receive Buffer Descriptor Entry

The receive buffer queue pointer register points to this data structure.


Receive Buffer Queue Pointer (MAC Register)
Receive Buffer 0
Receive Buffer 1
Receive Buffer *n*
(In memory)
Receive Buffer Descriptor List (in memory)

To create this list of buffers: (Note: M is number of Buffer Descriptor Words / 2, and will be 1 or 2 or 3 depending on 64 bit addressing and timestamp insertion modes being enabled)

1. Allocate a number (*N*) of buffers of X bytes in system memory, where X is the DMA buffer length programmed in the DMA Configuration register.
2. Allocate an area 8NxM bytes for the receive buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by GEM_GXL, i.e. bit 0 of word 0 set to 0.
3. Add an extra descriptor to the end of queue with its used bit set (bit 0 in word 0 set to 1). This last descriptor in the queue may also have its wrap bit set (bit 1 in word 0 set to 1) in addition to its used bit. When receive is enabled at least one entry in the buffer descriptor ring needs its used bit set so it is not sufficient to set the wrap bit of the last buffer in the queue without also setting its used bit. The GEM_GXL can now prefetch receive descriptors and the used bit will be used as an indication to the hardware that all available descriptors have been prefetched.
4. Write address of receive buffer descriptor list and control information to GEM_GXL register receive buffer queue pointer.
5. The receive circuits can then be enabled by writing to the address recognition registers and the network control register.

Transmit Buffer List

Transmit data is read from areas of data (the buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (Transmit Buffer Queue) is a sequence of descriptor entries as defined in the table Transmit Buffer Descriptor Entry – Non-LSO Frame

The transmit buffer queue pointer register points to this data structure.

To create this list of buffers: (Note: M is number of Buffer Descriptor Words / 2, and will be 1 or 2 or 3 depending on 64 bit addressing and timestamp insertion modes being enabled)

1. Allocate a number (N) of buffers of between 1 and 16383 bytes of data to be transmitted in system memory. Up to 128 buffers per frame are allowed.
2. Allocate an area 8NxM bytes for the transmit buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by GEM_GXL, i.e. bit 31 of word 1 set to 0.

3. Add an extra descriptor to the end of queue with its used bit set (bit 31 in word 1 set to 1). This last descriptor in the queue may also have its wrap bit set (bit 1 in word 0 set to 1) in addition to its used bit. When transmit is enabled at least one entry in the buffer descriptor ring must have its used bit set. When the buffer descriptor ring is initialized for the first time there must be a used bit set before or at the buffer descriptor with the wrap wrap bit. Once transmission halts due to reading a used bit firmware can reuse the transmit buffers and clear the used bits before and including the one with the wrap bit and restart transmission by writing the tx_start bit.
4. The GEM_GXL can now read transmit data so fast that all data may be read in before it sets the used bit of the first buffer descriptor in the queue.
5. Write address of transmit buffer descriptor list and control information to GEM_GXL register transmit buffer queue pointer.
6. The transmit circuits can then be enabled by writing to the network control register.

Address Matching

The GEM_GXL register pair hash address and the specific address register-pairs must be written with the required values. Each register pair comprises of a bottom register and top register with the bottom register being written first. The address matching is disabled for a particular register pair after the bottom register has been written and re-enabled when the top register is written. Each register pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

As an example, to set specific address register 1 to recognize destination address 21:43:65:87:A9:CB, the following values would be written to specific address register 1 bottom and specific address register 1 top:

Specific address register 1 bottom bits 31:0 (0x98): 8765_4321 hex.

Specific address register 1 top bits 15:0 (0x9C): cba9 hex.

PHY Maintenance

The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit two is set in the network status register (about 2000 `pclk` cycles later when bits [18:16] are set to 010 in the network configuration register). An interrupt is generated as this bit is set.

During this time, the MSB of the register is output on the `mdio_in` pin and the LSB updated from the `mdio_in` pin with each MDC cycle. This causes the transmission of a PHY management frame on MDIO. See section 22.2.4.5 of the IEEE 802.3 standard.

Reading during the shift operation will return the current contents of the shift register. At the end of the management operation the bits will have shifted back to their original locations. For a read operation the data bits will be updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

MDC should not toggle faster than 2.5 MHz (minimum period of 400 ns), as defined by the IEEE 802.3 standard. `mdc` is generated by dividing down `pclk`. Three bits in the network configuration register determine by how much `pclk` should be divided to produce MDC. The default is 32 which is acceptable for `pclk` running up to 80 MHz.

Interrupts

There are multiple interrupt conditions that are detected within the GEM_GXL. These are ORed to make a single interrupt (or multiple interrupts if priority queuing is enabled). Depending on the overall system design this may be passed through a further level of interrupt collection (interrupt controller). On receipt of the interrupt signal, the CPU shall enter the interrupt handler. Refer to your documentation for the interrupt controller to identify that it is the GEM_GXL that is generating the interrupt. To ascertain which interrupt, read the interrupt status register. Note that in the default configuration this register will clear itself after being read, though this may be configured to be write-one-to-clear if desired.

At reset all interrupts are disabled. To enable an interrupt, write to interrupt enable register with the pertinent interrupt bit set to 1. To disable an interrupt, write to interrupt disable register with the pertinent interrupt bit set to 1. To check whether an interrupt is enabled or disabled, read interrupt mask register: if the bit is set to 1, the interrupt is disabled.

Transmitting Frames

To set up a frame for transmission:

1. Enable transmit in the network control register.
2. Allocate an area of system memory for transmit data. This does not have to be contiguous, varying byte lengths can be used if they conclude on byte borders.
3. Set-up the transmit buffer list by writing buffer addresses to word zero of the transmit buffer descriptor entries and control and length to word one.
4. Write data for transmission into the buffers pointed to by the descriptors.
5. Write the address of the first buffer descriptor to transmit buffer descriptor queue pointer. When priority queues are used, each queue's buffer descriptor queue pointer should be updated and Q0's pointer should be updated last of all.
6. Enable appropriate interrupts.
7. Write to the transmit start bit in the network control register, or toggle the `trigger_dma_tx_start` pin

Receiving Frames

When a frame is received and the receive circuits are enabled, the GEM_GXL checks the address and, in the following cases, the frame is written to system memory:

If it matches one of the four specific address registers.

If it matches one of the four type ID registers.

If it matches the hash address function.

If it is a broadcast address (0xFFFFFFFFFFFF) and broadcasts are allowed.

If the GEM_GXL is configured to "copy all frames".

If a match is found in the external address filtering interface.

The register receive buffer queue pointer points to the next entry in the receive buffer descriptor list and the GEM_GXL uses this as the address in system memory to write the frame to.

Once the frame has been completely and successfully received and written to system memory, the GEM_GXL then updates the receive buffer descriptor entry (see the table Receive Buffer Descriptor Entry
) with the reason for the address match and marks the area as being owned by software. Once this is complete, a receive complete interrupt is set. Software is then responsible for copying the data to the application area and releasing the buffer (by writing the ownership bit back to 0).

If the GEM_GXL is unable to write the data at a rate to match the incoming frame, then a receive overrun interrupt is set. If there is no receive buffer available, i.e. the next buffer is still owned by software, a receive buffer not available interrupt is set. If the frame is not successfully received, a statistic register is incremented and the frame is discarded without informing software.

Section 3 - Integrating the GEM_GXL

Introduction

This section enables the GEM_GXL module to be fully integrated into an SoC design. It covers the following topics:

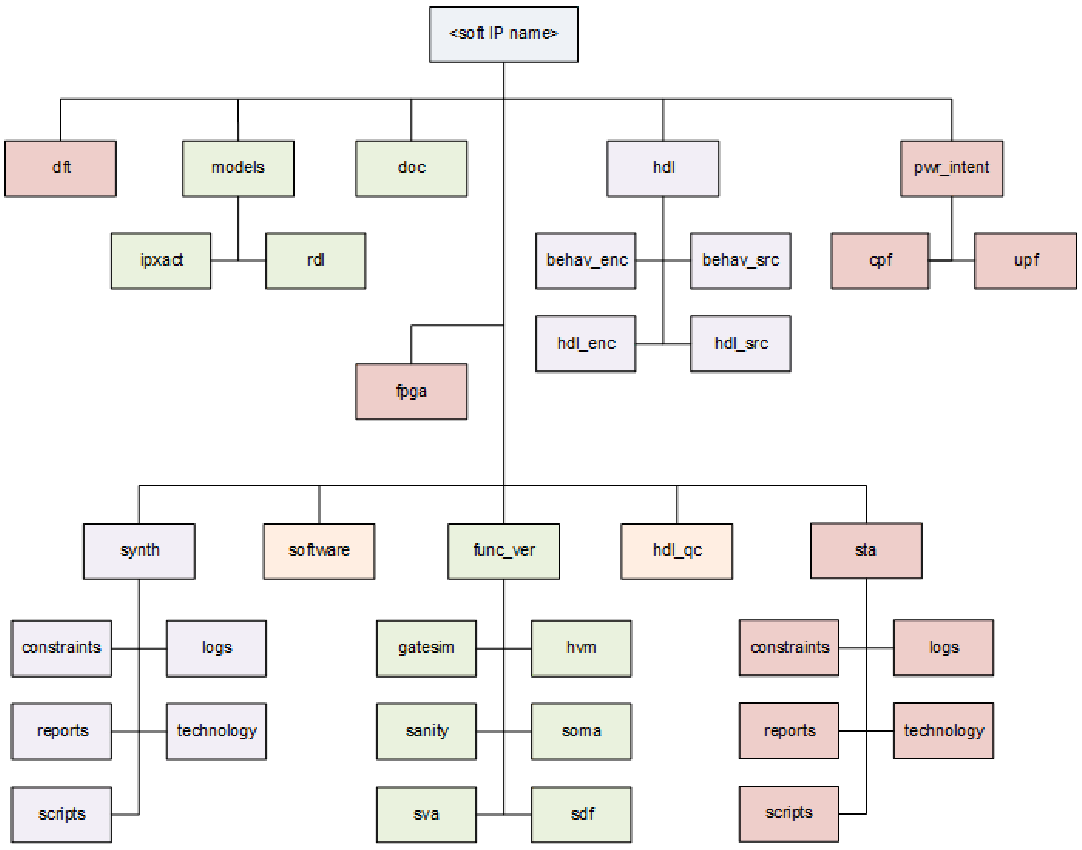
IP Content

Implementation Application Notes

Verification and Test Application

IP Content

The delivery directory structure for all IPG soft IP blocks is depicted in the diagram below. Colours in the diagram are intended only to make it easier to identify primary directories and the subdirectories that accompany them – no further relationship between content of similarly-shaded directories should be inferred. All soft IP deliveries originating from Cadence design IP business units will follow this structure, even if certain directories are not populated for a given IP block.



Customer delivery subdirectory contents are based on a set of directory definitions intended to accommodate each possible type of design view or data set that could generally apply to Cadence IP. Hard and soft IP deliveries each use a unique subset of these directories, with these subsets being common across all IPG business units. This means that hard and soft IP deliveries from all groups will have a consistent look and feel, however, not all directories are applicable for individual designs hence they are left intentionally empty.

The rtl is located in the hdl/hdl_src directory.

The SDC constraints are located in the synth/constraints directory.

In the case of the GEM_GXL the standalone Verilog testbench to demonstrate integration and basic traffic flow is located in the func_ver/hvm directory. The user may create their own work directory here and run the compile and simulate scripts.

Reference Environment

Tools

RTL Simulator:	Cadence Incisive Simulator 15.10
Code Coverage	Cadence Incisive Simulator 15.10
Logic & Test Synthesis	Cadence RTL Compiler 14.20
ATPG	Cadence Encounter Test ET 14.1.100
Perl	v5.6.1 built for i686-linux

Libraries

- Library tcbn28hpmbwp35hvtss0p81v125c
    - 28nm slow: 0.81V, 125C
    - 28nm fast: 0.99V, 0C

Implementation Application Notes

Parameterization

The GEM_GXL has many parameterisation options to configure the IP during compilation stage. This is achieved using Verilog `` `define `` compiler directives in an include file called `gem_gxl_defs.v`. To aid in debug, these configurations are readable through the APB address space starting from address 0x280 – Refer to the programming Interface section for further details.

The delivered database is provided with configuration examples as follows:-

The receive packet buffer memory should be clocked from hclk/aclk.

Configuration	Description
`gem_gxl_defs_default.v`	Default configuration including most of the available functionality such as DMA with internal FIFOs and PCS
`gem_gxl_defs_fifo.v`	External FIFO interface configuration, excluding DMA
`gem_gxl_defs_pbuf.v`	AHB DMA packet buffer configuration with no PCS
`gem_gxl_defs_pbuf_axi.v`	AXI DMA packet buffer configuration with no PCS
`gem_gxl_defs_pbuf_3qs_axi.v`	AXI multi_queue DMA packet buffer configuration
`gem_gxl_defs_pbuf_8qs_axi.v`	AXI multi_queue DMA packet buffer configuration

`gem_gxl_defs_pbuf_2qs.v`	AHB multi queue DMA packet buffer configuration
`gem_gxl_defs_pbuf_3qs.v`	AHB multi queue DMA packet buffer configuration
`gem_gxl_defs_pbuf_8qs.v`	AHB multi queue DMA packet buffer configuration
`gem_gxl_defs_large_pbuf.v`	Larger DMA packet buffer configuration for testing bigger jumbo frame size
`gem_gxl_defs_small_pbuf.v`	Smaller DMA packet buffer configuration for testing partial store and forward with small buffers
`gem_gxl_defs_minarea.v`	Lowest gate count configuration. No DMA , PCS, registers or TSU
`gem_gxl_defs_extra_large_pbuf_axi.v`	Extra large DMA packet buffer configuration for testing large external memory sizes
`gem_gxl_defs_pbuf_3qs_axi_128b`	AXI with a 128b dma bus width configuration
`gem_gxl_defs_pbuf_3qs_axi_128b_dpram.v`	Mismatched configuration of gem_dma_bus_width, tx/rx_pbuf_data and emac_bus_width with multiple queues and AXI
`gem_gxl_defs_pbuf_3qs_ahb_128b_dpram.v`	Mismatched configuration of gem_dma_bus_width, tx/rx_pbuf_data and emac_bus_width with multiple queues and AHB
`gem_gxl_defs_pbuf_3qs_cbs.v`	AHB multi queue DMA packet buffer configuration width credit based shaping
`gem_gxl_defs_pbuf_spram.v`	AHB DMA packet buffer configuration with single port memory
`gem_gxl_defs_small_pbuf_spram.v`	Smaller DMA packet buffer configuration for testing partial store and forward with small buffers using single port memory
`gem_gxl_defs_pbuf_axi_hp.v`	High performance AXI configuration
`gem_gxl_defs_small_pbuf_16qs_axi_hp.v`	High performance AXI configuration with a small packet buffer and 16 queues.
`gem_gxl_defs_fifo_soft_config.v`	DMA and FIFO configuration, using a software option to switch between the DMA or FIFO interface

The selected configuration file is automatically copied to `gem_gxl_defs.v` when the supplied `compile.pl` script is used with the `–cf <configuration>` command line option, where the configuration is named as `gem_defs_<configuration>.v`

The user should create their own `gem_gxl_defs.v` configuration file using the delivered versions as a template and place it in the appropriate include path for their own simulation tool.

The following sections detail the parameterisation options that are available by modifying the `gem_gxl_defs.v` file.

There is a configuration generator tool that has been created to help the user create a configuration file. This tool is GUI based and offers detailed descriptions on the options available and can automatically generate the configuration file and required file listings needed in order to simulate and synthesize the design with the appropriate options. It can also generate example test cases that you can simulate specific to you configuration (these are in addition to the example testcases already provided with the delivery that are targeted to example reference design configurations. This tool can also generate synthesis constraints specific to the configuration and simulation scripts. Please note that this tool is in alpha state. We are providing it now to offer the user a simple interface to help understand and create the necessary configuration files. Please do the following to run the tool …

- cd gem_gxl/func_ver/hvm
- mkdir work
- cd work
- ../scripts/gem_cfg_generator.pl

Common Configuration

Select whether to expose a GMII/MII or an RGMII interface. Comment the following if a GMII/MII interface is required. Uncomment for RGMII

```
//       `define gem_use_rgmii
```

Select whether to include an additional RMII interface. Uncomment the following to include a separate RMII port.

```
//       `define gem_include_rmii
```

The PCS module may be optionally omitted from the design by removing the double slash in the following statement:

```
//       ‘define gem_no_pcs
```

If using the PCS, define the interface option, the interface can be configured to be 'legacy' which is the interface defined in the IEEE 802.3 Clause 36 specifications, or a generic 10 or 20-bit interface. For the generic interface options, the PCS will also contain comma and code group alignment functions.
Only one of these options should be uncommented:
`define gem_pcs_legacy_if
//`define gem_pcs_10b_if
//`define gem_pcs_20b_if

Changing the following will remove the internal loop back functionality from the GEM. The internal loop back functionality requires gating of the `tx_clk` and `rx_clk`. This can be a problem for FPGA prototyping. If integrators are compiling for FPGA operation, they may find it convenient to remove the internal loop back functionality. Note that removing the loop back module will also remove the top level pins associated with it.

```
//       define gem_int_loopback for loopback capability
         `define gem_int_loopback

NOTE: Only one of the following defines can be active at one time:
gem_ext_fifo_interface
gem_tx_add_fifo_if
gem_host_if_soft_select
```

To use the exposed FIFO interface, remove the double slash in `gem_def.v` as follows:-

```
//       define gem_ext_fifo_interface
```

When using the external FIFO interface the DMA module is automatically omitted from the design.

To use the Low Latency TX interface, remove the double slash in `gem_def.v` as follows:-

```
//       define gem_tx_add_fifo_if
```

(Note: The low_latency_tx interface is mutually exclusive with the exposed FIFO interface i.e. only one can be compiled at a time. This is because the low latency TX interface uses the same ports as the external FIFO transmit interface)

When using the low_latency_tx interface the DMA is also in the design.

To be able to select External FIFO Interface OR DMA interface using a software control register, remove the double slash in `gem_def.v` as follows:-

```
//      define gem_host_if_soft_select
```

This define will build the deisgn with both external fifo interface port and DMA port. The user can then select which port to use by programming the enable_external_fifo_interface register

Note: this enable is not intended to be a dynamic selection and should be performed after the device has had a hard reset but before the data path is enabled using transmit enable and receive enable bits in the network control register.

The `gem_gxl_defs.v` file allows the number of user I/Os to be parameterised between 0 and 16 for both the user inputs and user outputs. Default is set to 8 for both inputs and outputs with the following statements:-

```
         `define gem_user_io
         `define gem_user_out_width 8
         `define gem_user_in_width 8
```

The GEM statistics registers may be optionally removed by ensuring that `gem_no_stats is defined.

```
//       `define gem_no_stats
```

To include the snapshot function in the statistics registers, add a double slash comment in the following statement. If this function is required, approximately 984 flip-flops and 10K gates are added to the implementation.

```
         'define gem_no_snapshot
```

The `` `gem_irq_read_clear `` define can be removed to cause the interrupt status register to use a write-one-to-clear mechanism rather than being cleared when read. If the following define is commented out then the interrupt status register can only be cleared by writing a '1' to the appropriate bit.

```
         `define gem_irq_read_clear
```

The maximum size of accepted jumbo frames can be controlled through the following define. In dma mode, this shouldnt be set any larger than 16383 due to the fact the descriptor writeback only has enough bits to register 16k in the length field.

```
         'define gem_jumbo_max_length       16'd10240
```

Filter Configuration

The number of specific address match registers is configurable via the following define. If no specific address filters are required, then leave the following define commented. However it is recommended that at least one is defined because the values of specific address 1 is used in pause frame generation. Otherwise specify the number required. Setting this define in pbuf DMA mode to a number greater than 4 will remove external address match from receive buffer descriptor entry word 1 and bit 28 of word 1 will be set if there is an address match in specific address register 1 to 8. Bits 27 to 25then indicate which specific address register matched. The first four filters are located at address offset 0x88 to 0xA4. Filters 5 to 36 are located between address offset 0x300 through 0x3fc.

```
`define num_spec_add_filters 4
```

The depth of the MAC receive pipeline can be configured using the following parameter:-

```
         `define gem_rx_pipeline_delay 10
```

It is recommended that this value is left unchanged at 10. This is because the design has been thoroughly verified with a value of 10. The only purpose of the receive pipeline is to delay frame reception until the frame filtering process completes. If an external filter that looks deep into a frame is used, a deeper pipeline may be required. If very low latency and only simple frame filtering is required users might contemplate using a shallower pipeline. When using internal address matching and DMA this must be a minimum of seven. Frames that are not recognized in time will be dropped.

Data path widths

Configure the MAC bus width (Used in conjunction with DMA configuration below)

1. For non DMA modes of operation the gem_emac_bus_width parameter defines the width of the tx_r_data and rx_w_data buses - i.e. the FIFO interface buses.

2. For Packet Buffer DMA operation the gem_emac_bus_width paramter should be set to the same size as the gem_dma_bus_width parameter, unless gem_dma_bus_width is set to 128. If gem_dma_bus_width is set to 128 then gem_emac_bus_width should be set to 64.

3. For Non Packet Buffer DMA operation gem_emac_bus_width should be set to ..._dma_bus_width.

The following table details standard configurations:

Configuration	_dma_bus_width	_t/rx_pbuf_data	_emac_bus_width	Notes
Packet buffer DMA (use ..._tx_pkt_buffer & ..._rx_pkt_buffer)	32	32	32	
	64	64	64	
	128	128	64	
Packet buffer DMA with SPRAM (use gem_spram)	32	128	32	
	64	128	64	
	128	Not Applicable	Not Applicable	Not Supported
Non packet buffer DMA	32	32	32	
	64	64	64	
	128	128	128	
External FIFO interface only	Not Applicable	Not Applicable	32	
	Not Applicable	Not Applicable	64	
	Not Applicable	Not Applicable	128	
Packet buffer DMA with additional external FIFO IF (use gem_tx_add_fifo_if)	64	128	32	More complex config with external fifo IF width as 32b and ahb/axi width as 64b

Note. When using AXI mode with a single port ram (gem_spram == 1) mode and a 32b dma bus width (gem_dma_bus_width == 32 or bits 22:21 of the network_config register are set to 0) the following hidden registers need to be updated (these registers are used for fine tuning AXI dma bursts and normally should not be touched):

- Write 0x4f8 to register 0x002A010A.
- Write 0x4fc to register 0x0010001C.

DMA Configuration

The following defines are general DMA configuration - they apply to either GEM DMA implementations

The choice of whether to use an AXI or an AHB front end interface is determined by setting the axi define.

```
// Define if DMA should use native AXI or AHB.   Set the define for AXI.
`define gem_axi
```

If an AXI DMA is selected, the user may choose the level of buffering in the DMA

Buffering is required to improve AXI performance, allow a greater degree of AXI pipelining (where pipelining here refers to the number of AR or AW/W requests issued before the accompanying R or B response is returned from the AXI fabric).

The following define sets the number of bits used to describe the depth of the AXI pipelining allowed. A value of 1 means the maximum level of AXI pipelining is 2. A value of 4 means the maximum level of AXI pipelining is 16. Maximum is 4.

Note the degree of pipelining can also be fine tuned using a programmable register in the core.

```
  `define gem_axi_access_pipeline_bits 4'h4
```

The following define sets the number of bits used to describe the depth of the FIFO that is needed to buffer TX descriptor read responses. Minimum value is 1. Max tested value is 4. A higher value improves the ability of the DMA and MAC to maintain maximum performance in high latency systems.

```
  `define gem_axi_tx_descr_rd_buff_bits 4'h4
```

The following define sets the number of bits used to describe the depth of the FIFO that is needed to buffer RX descriptor read responses. Minimum value is 1. Max tested value is 4. A higher value improves the ability of the DMA and MAC to maintain maximum performance

```
  `define gem_axi_rx_descr_rd_buff_bits 4'h4
```

The following define sets the number of bits used to describe the depth of the FIFO that is needed to buffer TX descriptor write requests. This buffer is needed to avoid the underlying TX DMA from being blocked from continuing transmission as a result of a large RX data write in progress. Minimum value is 1. Max tested value is 4. A higher value improves the ability of the DMA and MAC to maintain maximum performance

```
  `define gem_axi_tx_descr_wr_buff_bits 4'h4
```

The following define sets the number of bits used to describe the depth of the FIFO that is needed to buffer RX descriptor write requests. It is needed primarily for high latency systems. The DMA has the ability to issue multiple write requests and data before the first response (on B channel) has returned. Since the descriptor write must not be issued until the fabric identifies the last data write of a packet has been successfully written, and the underlying DMA issues these at request phase time, the DMA must buffer these descriptor writes.Minimum value is 1. Max tested value is 4. A higher value improves the ability of the DMA and MAC to maintain maximum performance in high latency systems.

`define gem_axi_rx_descr_wr_buff_bits 4'h4

The maximum width of DMA AHB/AXI bus can be configured using the `` `..._dma_bus_width `` define. Valid settings are 32, 64 or 128. Reducing the bus width will save area resources. The current bus width can be programmed through the network configuration register but will be forced to a value no greater than the configured width

```
         `define gem_dma_bus_width        128
```

The maximum width of DMA AHB/AXI address bus can be configured to 32 bits or 64 bits using the `` `gem_dma_addr_width `` define. Valid settings are 32 and 64

The value for hprot AHB output can be controlled using `` `..._hprot_value ``. Individual bits assigned as follows:

```
//        bit  |   set to 1    |    set to 0
//     --------------------------------------
//     hprot[0] |  data access  |  opcode access
//     hprot[1] |  privileged   |  user
//     hprot[2] |  bufferable   |  not bufferable
//     hprot[3] |  cacheable    |  not cacheable
   `define gem_hprot_value        4'b0001
```

The value for arprot and awprot AXI output is controlled via `` `…_axi_prot_value ``. Defaulted to nonsecure and data access. Individual bits are assigned as follows:

```
//        bit  |   set to 1    |    set to 0
//     --------------------------------------
//     prot[0] |  privileged   |  normal
//     prot[1] |  nonsecure    |  secure
//     prot[2] |  instruction  |  data access
   `define gem_axi_prot_value     3'b010
```

The value for arcache and awcache AXI output via `` `gem_axi_cache_value ``. Defaulted to Non-cacheable and nonbufferable

```
   `define gem_axi_cache_value     4'b0000
```

The following defines are specific to the PACKET BUFFER DMA implementation

The DMA can be configured to use low latency internal FIFOs implemented using synthesizable flip-flops or to utilise the packet buffer with external DPRAM. To use the packet buffer configuration ensure `` `gem_rx_pkt_buffer `` and `` `gem_tx_pkt_buffer `` are defined. The packet buffer configuration should not be set if using the exposed FIFO interface.

```
//       `define gem_rx_pkt_buffer
//       `define gem_tx_pkt_buffer
```

Packet Buffer mode of operation has the option of using a SPRAM, rather than the default DPRAM. If this option is used there are frequency limitations where the AHB/AXI clock must be going 2x faster than the MAC data rate.

`define gem_spram

If the DMA is configured to use packet buffers, the maximum size of external DPRAM can be configured using the following defines. Maximum number allowed is 16 (represents 256kKBytes with 32 bit datapath):-

```
        `define gem_rx_pbuf_addr  11 // RX buffer depth = 2 ^ gem_rx_pbuf_addr
        `define gem_tx_pbuf_addr  10 // TX buffer depth = 2 ^ gem_tx_pbuf_addr
```

Define the size of the TX packet buffer SRAM data width if gem_tx_pkt_buffer is defined. Valid settings 32, 64 and 128. Note. The SRAM data width cannot be less than `` `gem_dma_bus_width ``

```
        `define gem_tx_pbuf_data             128
```

Define the size of the RX packet buffer SRAM data width if gem_rx_pkt_buffer is defined. Valid settings 32, 64 and 128. Note. The SRAM data width cannot be less than `` `gem_dma_bus_width ``

```
         `define gem_rx_pbuf_data             128
```

The DMA can support partial store and forward mode. If partial store and forward mode operation is required then this define should be set. Partial store and forward mode facilitates lower latency through the packet buffer and also allows frames larger than the buffer size to be supported. Partial store and forward mode needs additional FIFOs and if there is no plan to use partial store and forward mode then by not setting this define there is an area saving of around 2k gates. If this define is set then partial store and forward mode additionally needs a software enable by setting the appropriate enables in the cutthru registers.

```
  `define gem_pbuf_cutthru
```

The DMA can support priority queues. The number of queues to support is configurable using the following defines. Undefine all of these to disable priority queuing.

```
   `define dma_priority_queue1
   `define dma_priority_queue2
   `define dma_priority_queue3
   `define dma_priority_queue4
   `define dma_priority_queue5
   `define dma_priority_queue6
   `define dma_priority_queue7
   `define dma_priority_queue8
   `define dma_priority_queue9
   `define dma_priority_queue10
   `define dma_priority_queue11
   `define dma_priority_queue12
   `define dma_priority_queue13
   `define dma_priority_queue14
   `define dma_priority_queue15
```

The lowest priority queue will be queue0. For each transmit queue (starting from queue0), there is an associated transmit DPRAM region and the address space allocated to it will be defined as follows:

The total DPRAM space will be configured using the existing define `gem_tx_pbuf_addr.

To support priority queues, this space will be split into equal sized segments. The size of the segments are configurable using the define `` `gem_tx_pbuf_queue_segment_size `` . `` `gem_tx_pbuf_queue_segment_size `` only needs to be set if priority queining is enabled and defines how many of the DPRAM address bits specified in `` `gem_tx_pbuf_addr `` will be used for the number of segments.

Eg. If `` `gem_tx_pbuf_queue_segment_size `` = 4, then there are 16 segments each holding 1/16 of the total DPRAM space.

If `` `gem_tx_pbuf_queue_segment_size `` = 1, then there are 2 segments each holding 1/2 of the total DPRAM space.

One further define per active queue is needed to allocate the number of segments required by each queue. This is achieved by defining the number of bits to ensure a power of two is always maintained. Note that if only 1 segment is required for a queue, then the define associated with that queue should not be set.

Eg. For 4 queues, with queue0 using a quarter of the TX DPRAM space, queue1 and queue2 each using an eighth and queue3 using half, the defines would be setup as follows :

```
`define dma_priority_queue1
`define dma_priority_queue2
`define dma_priority_queue3
`define gem_tx_pbuf_queue_segment_size      3 // Eight segments
`define gem_tx_pbuf_num_segments_q0         1 // Queue 0 uses 2 segments
`define gem_tx_pbuf_num_segments_q3         2 // Queue 1 uses 4 segments
```

`gem_tx_pbuf_num_segments_q1 and `gem_tx_pbuf_num_segments_q2 are not defined as they only use 1 segment

The RX priority queue screening algorithm in the receive direction used to identify receive queue information is enabled and configured using the following defines. Do not define either of the following if you do not wish to include any screeners, otherwise compilation errors will occur.

```
  `define num_type1_screeners 8'd16   // Number of screener type 1 registers
  `define num_type2_screeners 8'd16   // Number of screener type 2 registers
```

If the num_type2_screeners define has been set, the number of ethertype match registers will need to be defined up to a maximum of 8. For zero ethertype match registers (to minimize area), then do NOT define num_scr2_ethtype_regs at all.

```
  `define num_scr2_ethtype_regs 8'd8   // Number of ethertype match registers
```

If the num_type2_screeners define has been set, the number of field compare match registers will need to be defined up to a maximum of 32. For zero field compare match registers(to minimize area), then do NOT define num_scr2_compare_regs at all.

```
  `define num_scr2_compare_regs 8'd16  // Number of field compare match registers
```

When multiple transmit queues are present, choose whether to include the credit based shaper algorithm for the top 2 queues. A credit-based shaping algorithm is available on the two highest priority queues and is defined in 802.1Qav: Forwarding and Queuing Enhancements for Time-Sensitive Streams. This allows traffic on these queues to be limited and to allow other queues to transmit. this define is negative. So only uncomment if you do NOT want the CBS logic

```
//   `define gem_exclude_cbs
```

The following defines are specific to the FIFO based DMA implementation. The following defines are ignored if the `gem_rx_pkt_buffer is defined

If the DMA is configured to use internal FIFOs, these statements set the size of the FIFOs:

```
         `define gem_rx_fifo_size 10
         `define gem_rx_base2_fifo_size 4’b1010
         `define gem_rx_fifo_cnt_width 4

         `define gem_tx_fifo_size 10
         `define gem_tx_base2_fifo_size 4’b1010
         `define gem_tx_fifo_cnt_width 4
```

This will set the receive FIFO and transmit FIFO word depths to 10. The GEM has been verified with these parameters set to 10.

The use of any other sizes would require verification and possible consequential code changes.

TSU inclusion

If TSU enhancements for 1588 support are not required comment the following

```
       `define gem_tsu
```

If External TSU port is not used comment the following
```
       `define gem_ext_tsu_timer
```

Clock TSU from tsu_clk rather than pclk
```
       `define gem_tsu_clk
```

Transmit and Receive Offload inclusion

If Large Send Offload is not required comment the following
```
       `define gem_pbuf_lso
```
If Receive Side Coalescing is not required comment the following
```
       `define gem_pbuf_rsc
```

Revision Register Values

The GEM contains readable 16-bit fields for part number and revision number, where the part number is fixed to 0x0007 for GEM_GXL and the revision register is incremented after each major release of the IP.

The hard wired module revision register value is set using the following parameter:
```
         `define gem_revision_reg_value 32’hxxxxxxxx
```

The hard wired value of the PCS PHY identification registers can be configured using the following statements:
```
         `define gem_phy_id_top 16’h0002
         `define gem_phy_id_bot 16’h010b
```

In the PCS module, the PCS identification register may be optionally removed by commenting out the following line in the `gem_gxl_defs.v` file:
```
         `define gem_phy_ident
```

Default Register Values

The default values for some programmable registers can be defined at compile time, though they are still over-writeable by software.

The default DMA bus width can be set in the network configuration register.
```
// 2'b00 - 32 bits
// 2'b01 - 64 bits
// 2'b1x - 128 bits
   `define gem_dma_bus_width_def 2'b00
```

The amount PCLK should be divided to generate MDC can be set in the network configuration register as follows:-
```
// 3'b000 - divide pclk by 8
// 3'b001 - divide pclk by 16
// 3'b010 - divide pclk by 32
// 3'b011 - divide pclk by 48
// 3'b100 - divide pclk by 64
// 3'b101 - divide pclk by 96
// 3'b110 - divide pclk by 128
// 3'b111 - divide pclk by 224
   `define gem_mdc_clock_div 3'b010
```

The default DMA endianism value in DMA control register can be set as follows:-
```
//             packet data             management descriptors
// 2'b00 -    little endian               little endian
// 2'b01 -    little endian                  big endian
// 2'b10 -       big endian               little endian
// 2'b11 -       big endian                  big endian
`define gem_endian_swap_def 2'b00
```

The default RX DMA packet buffer memory size in DMA configuration register can be set as follows:-

```
// 2'b11 - use full configured memory size
// 2'b10 - use half of configured memory size
// 2'b01 - use quarter of configured memory size
// 2'b00 - use eighth of configured memory size
`define gem_rx_pbuf_size_def 2'b11
```

The default TX DMA packet buffer memory size in DMA configuration register can be set as follows:-

```
// 1'b1 - use full configured memory size
// 1'b0 - use half of configured memory size
`define gem_tx_pbuf_size_def 1'b1
```

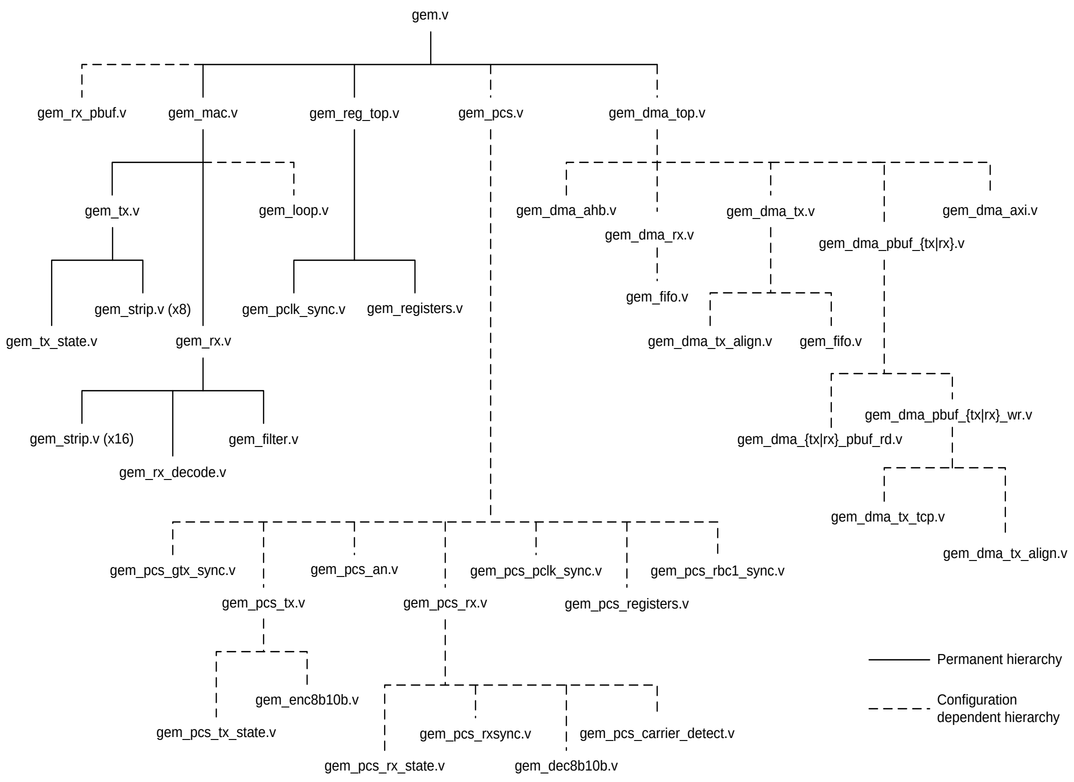
The default value of the external DMA receiver buffers are set to 128 byte with the following code and must be a multiple of 64 bytes up to 16320 bytes:

```
‘define gem_rx_buffer_length_def 8’d2
```

Other Configuration Parameters

In `gem_gxl_defs.v`, the APB address map is defined with macro definitions for the address locations.

Design Hierarchy



IP Application

APB Interface

The APB is a standard AMBA APB interface as described in ARM Document IHI 0011A, AMBA Specification (Rev 2.0).

When instantiating this IP, please refer to the User Guide for your APB Decoder for instructions how to define the address decode for generating `psel`. The GEM_GXL module requires `psel` to be decoded from `paddr[31:12]` and requires `paddr[11:2]` directly. `paddr[1:0]` are ignored because the GEM register map consists only of 32 bit registers.

AHB Master Interface

The AHB Master is a standard AMBA AHB interface as described in ARM Document IHI 0011A, AMBA Specification (Rev 2.0). Split and retry operations are not supported.

AXI Master Interface

The AXI Master is a standard AMBA AXI4 interface as described in the latest ARM AMBA specifications.

MII/GMII

The core connects to external PHYs using the GMII/MII interface (`txd`, `tx_er`, `tx_clk`, `tx_en`, `rxd`, `rx_er`, `rx_clk`, `rx_dv`, `col` and `crs`). Configuration and status information is exchanged between the GEM and PHYs using the management interface (MDC and MDIO).

RGMII

The RGMII specification is available on the web at this location:
http://www.hp.com/rnd/pdfs/RGMIIv2_0

In RGMII mode, the core receives a data nibble on rgmii_rxd[3:0] on both the edges of the rx_clk along with a single bit control signal. The lower nibble of the data byte is received on the rising edge of rx_clk, and the upper nibble is received on the falling edge. The RGMII module receives a control bit on rgmii_rx_ctl on both edges of rx_clk. This control bit contains the data valid signal `gmii_rx_dv` and encoded receiver error information on `gmii_rx_er`. The RGMII module also uses the data and control signals to extract the in-band PHY status for carrier sense, collision, speed, link and duplex.

The RGMII module is implemented with positive clock edge flip-flops. The `n_rx_clk` clock input is used to sample `rgmii_rxd` on the negative edge of `rx_clk`.

In-band Status Decoding in RGMII Mode

The status of the PHY is sampled and decoded from `rgmii_rxd[3:0]` on the rising edge of the `rx_clk`, whenever normal data, data error, carrier extend, carrier sense, or false carrier are not present. This is indicated by `rgmii_tx_ctl` being sampled low on both the rising and falling edges of `rx_clk`. In-band status is decoded as shown in the following table.

Status	**Range of rgmii_rxd[3:0] Decoded on Rising Edge**	**Description**
`rgmii_link_status`	`rgmii_rxd[0]`	Indicates link status 0 = down 1 = up

`rgmii_gmii_speed[1:0]`	`rgmii_rxd[2:1]`	Indicates receive clock speed 00 = 2.5 MHz 01 = 25.0 MHz 10 = 125.0 MHz 11 = reserved
`rgmii_gmii_duplex_out`	`rgmii_rxd[3]`	Indicates duplex status 0 = half duplex 1 = full duplex

When `gmii_tx_en` is asserted or a valid carrier sense is decoded from the received control and status. A valid carrier sense is triggered when a false carrier, carrier extend or carrier sense error is detected.

The GEM transmits the lower nibble of the data byte on `rgmii_txd[3:0]` on the rising edge of `rgmii_tx_clk` and the upper nibble is transmitted on the following falling edge.

`rgmii_tx_clk_sig` selects the lower nibble to be output on `rgmii_txd` when high and the upper nibble when low so needs to be a slightly delayed version of `rgmii_tx_clk`.

Control signals `gmii_tx_en` and `gmii_tx_er` are multiplexed on `rgmii_tx_ctl` in a single clock cycle using a similar technique.

From release 1p09 onwards it is possible to configure the RGMII interface for MII operation by using software to set bit 28 in the network control register. If this bit is set rgmii_tx_ctl is equivalent to tx_en, rgmii_rx_ctl is equivalent to rx_dv and the rx_er and tx_er top-level signals become active. From release 1p09 onwards the col, crs, rx_er and tx_er top level signals are brought out when `define gem_use_rgmii is set so half-duplex MII operation and LPI signalling may be supported. The user may leave tx_er unconnected and drive col, crs and rx_er low if they do not want to use these pins. The col and crs signals are only needed for half-duplex operation. The tx_er and rx_er signals are required if LPI signalling is desired over MII. The rx_er signal is also used to indicate errors detected by the attached PHY when configured for MII operation.

RMII

When configured for RMII operation the tx_clk timed outputs from the GEM, txd and tx_en, are clocked into the ref_clk clock domain of the RMII interface. In implementing the tx_clk and ref_clk trees care must be taken to ensure that there can be no setup and hold violations on txd and tx_en when they are sampled in the ref_clk domain.

The ref_clk timed outputs from the RMII interface, rx_er, rx_dv and rxd, are clocked into the rx_clk domain of the GEM. To prevent hold violations rxd and rx_dv are designed to transition coincident with the falling edge of the rmii_rx_clk output of the RMII interface. To achieve this the rmii_rx_clk time may be stretched by 20 ns (ie by one ref_clk period). In implementing the rx_clk and ref_clk trees care must be taken to ensure that there can be no setup and hold violations on rxd and rx_dv when they are sampled in the rx_clk domain.

`rx_er` can change on either edge of `ref_clk` but remains valid until the end of frame. Therefore `rx_er` can be treated as a multicycle path and hold errors on this signal going into the `rx_clk` clock domain can be ignored.

Any setup and hold violations on col and crs going into the tx_clk and rx_clk domains can be ignored.

MDIO

`mdio` is a single bi-directional tristate signal going between the GEM and one or more PHYs. The GEM signals `mdio_in`, `mdio_out` and `mdio_en` are provided to control a chip-level tri-state buffer. Alternatively `mdio` may be implemented as a pull-down. In this case the enable to the pull-down should be driven with `~mdio_out & mdio_en.`

Other Interface Signals

The `ext_interrupt_in` signal is an optional GEM input that can be used as an interrupt from the PHY or elsewhere. This signal is redundant when using the management interface as the PHY's status can be polled using `mdc` and `mdio` which are controlled through the PHY maintenance register.

Alternatively the GEM can communicate with PHYs or SERDES using the TBI interface. The GEM contains built-in PCS functionality so it can support 1000BASE-SX and 1000BASE-LX fibre optic applications using its TBI interface in conjunction with a SERDES and appropriate optical module.

The TBI, GMII and MII are standard interfaces and the GEM will inter-operate with any PHY that supports these interfaces.

Clock Requirements

Clocks

Depending on configuration the GEM has multiple clock domains as follows:

`hclk` (10 to 500 MHz): AMBA AHB clock used within DMA_TOP for both transmit and receive DMA operation.

`aclk` (10 to 500 MHz): AMBA AXI clock used within DMA for both transmit and receive DMA operation.

`pclk` (10 to 500 MHz): AMBA APB clock used within REG_TOP and PCS register block. For data-rates of 2.5G and above the minimum frequency for pclk is 20 MHz.

`tsu_clk` (5 to 400 MHz): Alternate clock source for the time stamp unit. Timestamp accuracy improves with higher frequencies. To support single step timestamping, tsu_clk frequency must be greater than 1/8th the frequency of tx_clk or rx_clk.

`tx_clk` (2.5, 25, 125 MHz): MAC transmit clock, used within GEM_MAC (`gem_tx`, `gem_dma_tx_pbuf_rd` and `gem_tx_state` blocks). In RMII mode, `tx_clk` should be connected to the `rmii_tx_clk` output. Otherwise, in 10/100 data rate mode `tx_clk` will run at either 2.5 MHz or 25 MHz as determined by the external PHY MII clock input. When using gigabit mode, the transmit clock must be sourced from a 125 MHz reference clock. Depending upon system architecture, this reference clock may be sourced from an on-chip clock multiplier, generated directly from an off-chip oscillator, or taken from the PHY `rx_clk`. In SGMII mode this clock is sourced from `gtx_clk`. In 100Mb/s SGMII mode `gtx_clk` is divided down to bring the effective frequency down to 12.5MHz and in 10Mb/s SGMII down to 1.25MHz.

`gtx_clk` (125/312.5 MHz) PCS transmit clock. In non-SGMII application this will be the same as `tx_clk`. In SGMII applications `gtx_clk` is at 125 or 312.5 MHz and divided down in 10 and 100Mb/s modes to drive `tx_clk`

`gtx20_clk` (62.5/156.25 MHz) PCS transmit clock to support 20-bit PHY interface. This must be phase aligned with gtx_clk.

`rx_clk` (2.5, 25, 62.5, 125 or 156.25 MHz): Clock used within GEM_MAC (`gem_rx`, `gem_rx_decode, gem_dma_pbuf_rx_wr` and `gem_filter` blocks). In RMII mode, `rx_clk` should be connected to the `rmii_rx_clk` output. Otherwise, in 10/100 and gigabit modes using the GMII/MII interface, this clock is sourced from the `rx_clk` input of the external PHY and will be either 2.5, 25 or 125 MHz. When the PCS is selected in gigabit mode, the clock must be sourced from either `rbc1` or pcs_rx_clk input and will be 62.5 or 156.25MHz. In 100Mb/s SGMII mode the clock is divided down to bring the effective frequency down to 12.5MHz and in 10Mb/s SGMII down to 1.25MHz. In 10/100 SGMII 8 bits of data are transferred from the PCS to the MAC each clock cycle rather than 16.

`rbc0` and `rbc1` (two 62.5 MHz clocks 180° out of phase): Clocks used in the PCS receive channel. If an on chip SerDes is present these should be generated by deleting alternate phases of `pma_rx_clk`. `rbc0` and `rbc1` alignment is achieved using the `rbc_align` output from the comma alignment block.

`pcs_rx_clk` (62.5 or 156.25MHz) PCS receive datapath clock used when gem_pcs_legacy_if is not defined.

`pcs_rx10_clk` (125 or 312.5MHz) PCS receive interface clock used when gem_pcs_legacy_if is not defined and supporting a 10-bit PHY interface.

`n_tx_clk` (2.5, 25 or 125 MHz): Inverted `tx_clk` used for loop back module and RGMII modes. For loopback, `tx_clk` domain signals are re-timed to this clock before being passed to the `rx_clk` domain receiver inputs.

`n_rx_clk` (2.5, 25 or 125 MHz): Inverted `rx_clk` used for RGMII modes.

The GEM contains synchronizers that ensure reliable propagation of signals across clock boundaries. Most clocks run asynchronously with respect to each other with data transfers implemented using handshaking logic. In tbi mode rbc1 (or pcs_rx_clk) and rx_clk need to run synchronously and in loopback mode the transmit and receive path need to be synchronous to each other.

As far as the GEM is concerned, MDC (management data clock) is not a clock, but a `pclk` timed output from the registers block.

The maximum frequency of `hclk/aclk` and `pclk` is determined by the speed of the technology library.

The `hclk/aclk` clock frequency must be chosen such that the bandwidth requirements of the chosen transmit and receive Ethernet data rates are met.

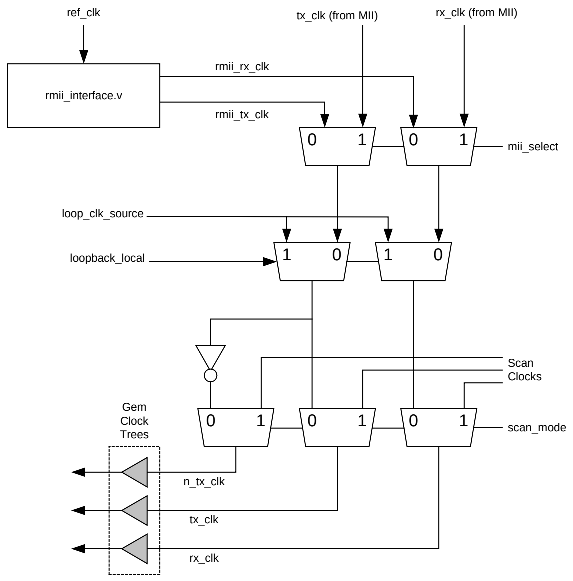
When using internal loop back mode, `tx_clk` and `rx_clk` must be provided using the same loop back reference clock, and `n_tx_clk` must be provided with the inverted version of the loop back reference clock.

It is important that receive and transmit are disabled when making the switch into and out of internal loop back, as the clocks provided may glitch whilst switching to the loop back reference clock.

When operating at gigabit speed using GMII interface, the system must provide a 125 MHz reference clock to the PHY, normally termed `g_tx_clk.` This clock must be the same clock used for the `tx_clk` input of the GEM to maintain correct relationship between the reference clock and data on the GMII.

RMII Clocking

When operating in RMII mode, both the `tx_clk` and `rx_clk` are internally generated from `ref_clk`. In order to provide maximum flexibility to the user, these generated clocks are fed out and then back into the design. This allows the user to determine their own scan insertion technique and helps with clock tree insertion and balancing. These signals can optionally be multiplexed with an external clock to provide clocking in internal loopback mode. `gem_clk_ctrl.v` gives an example of how this can be done, but it is also described below:



RGMII clocking

The RGMII standard specifies a source synchronous clock. It relies on the clock having a longer path delay than the data so that the data is resampled using the same edge of the clock on which it was generated. The RGMII standard supports two different versions. Version 1.3 relies on a PCB trace delay of 1.5 to 2.0 ns for the clock, whereas version 2.0 and later offers the option of introducing the delay on-chip at the source. Devices which support the internal delay (ID) are referred to as RGMII-ID.

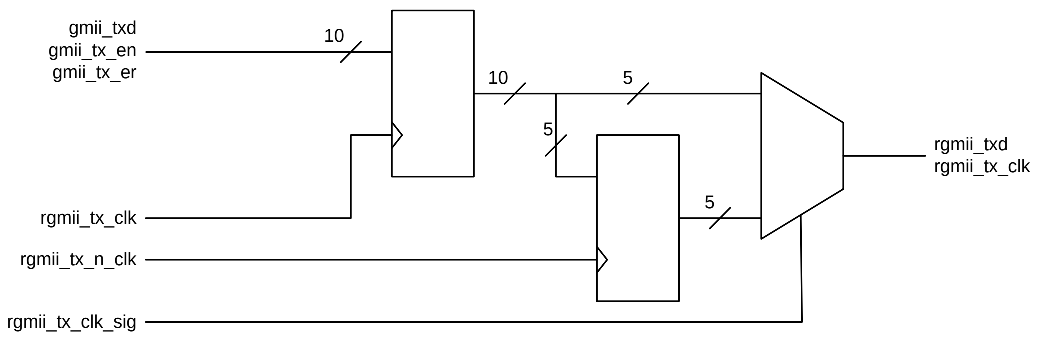
The other main difference between version 1.3 and 2.0 is the type of pads used. Version 2.0 utilises 1.5 V HSTL pads, whereas version 1.3 uses 2.5 V CMOS pads.

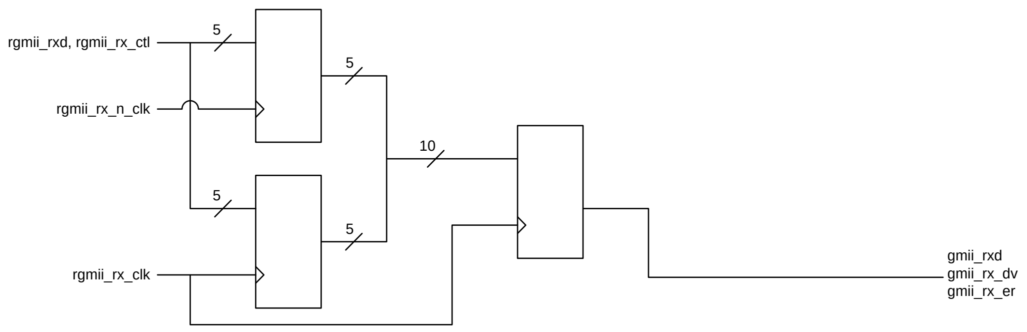
This IP supports both version of the RGMII spec but it is the responsibility of the implementer to insert the appropriate delay for RGMII-ID.

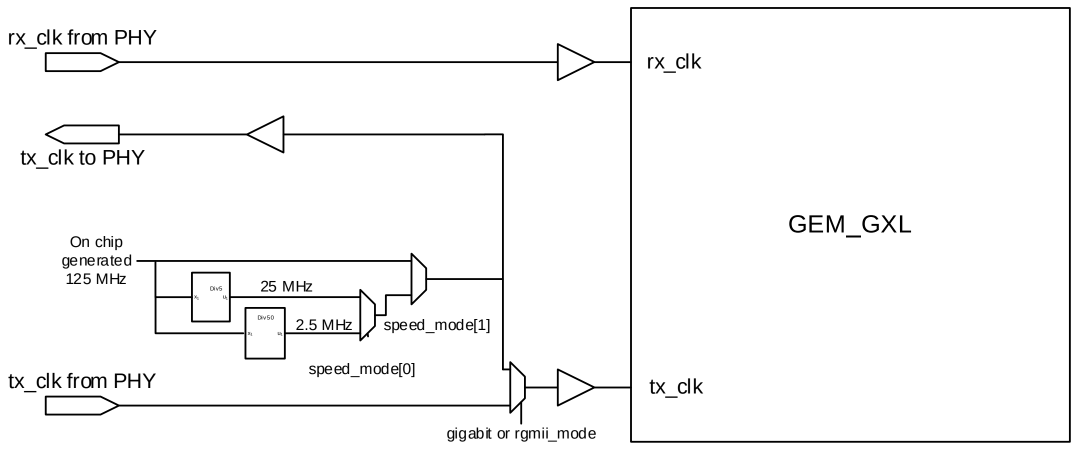
The RGMII module does not generate the `rgmii_tx_clk` signal. This must be generated by the system clock controller, and provided to both the RGMII module and to the PHY. To ensure that the skew requirements of the RGMII are met, for RGMII version 1.3 the delays on this external clock output should be matched to the RGMII transmit data and control outputs, for RGMII-ID the transmit clock output should be delayed 2ns relative to these outputs.

The RGMII uses the transmit clock to control the multiplexer logic in addition to its use as a real clock. When this clock is being used as a data signal, it is sourced from `rgmii_tx_clk_sig`. `rgmii_tx_clk_sig` should be a slightly delayed version of `rgmii_tx_clk`.

In the diagram below showing RGMII transmit operation the gmii_txd[7:0] data signals are sampled on the rising edge of rgmii_tx_clk. The lower nibble of the data byte rgmii_txd[3:0] is available on the rising edge of rgmii_tx_clk and the upper nibble is available on the following falling edge of rgmii_tx_clk. The signal rgmii_tx_clk_sig selects the lower nibble to be output on rgmii_txd when high and the upper nibble when low. The rgmii_tx_clk_sig signal needs to be a delayed version of rgmii_tx_clk so that the relevant input to the mux driving rgmii_txd is stable when rgmii_tx_clk_sig changes.





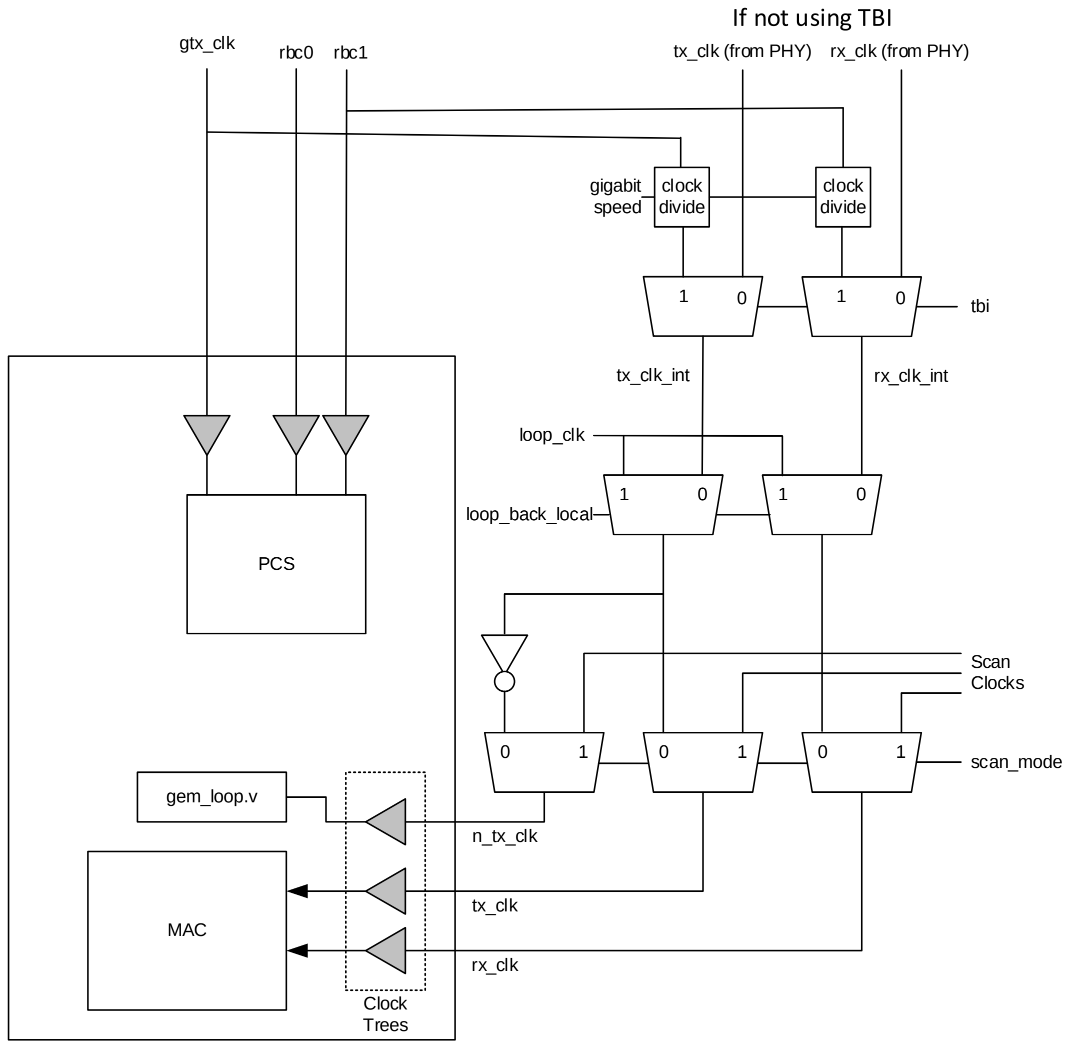


PCS Clocking

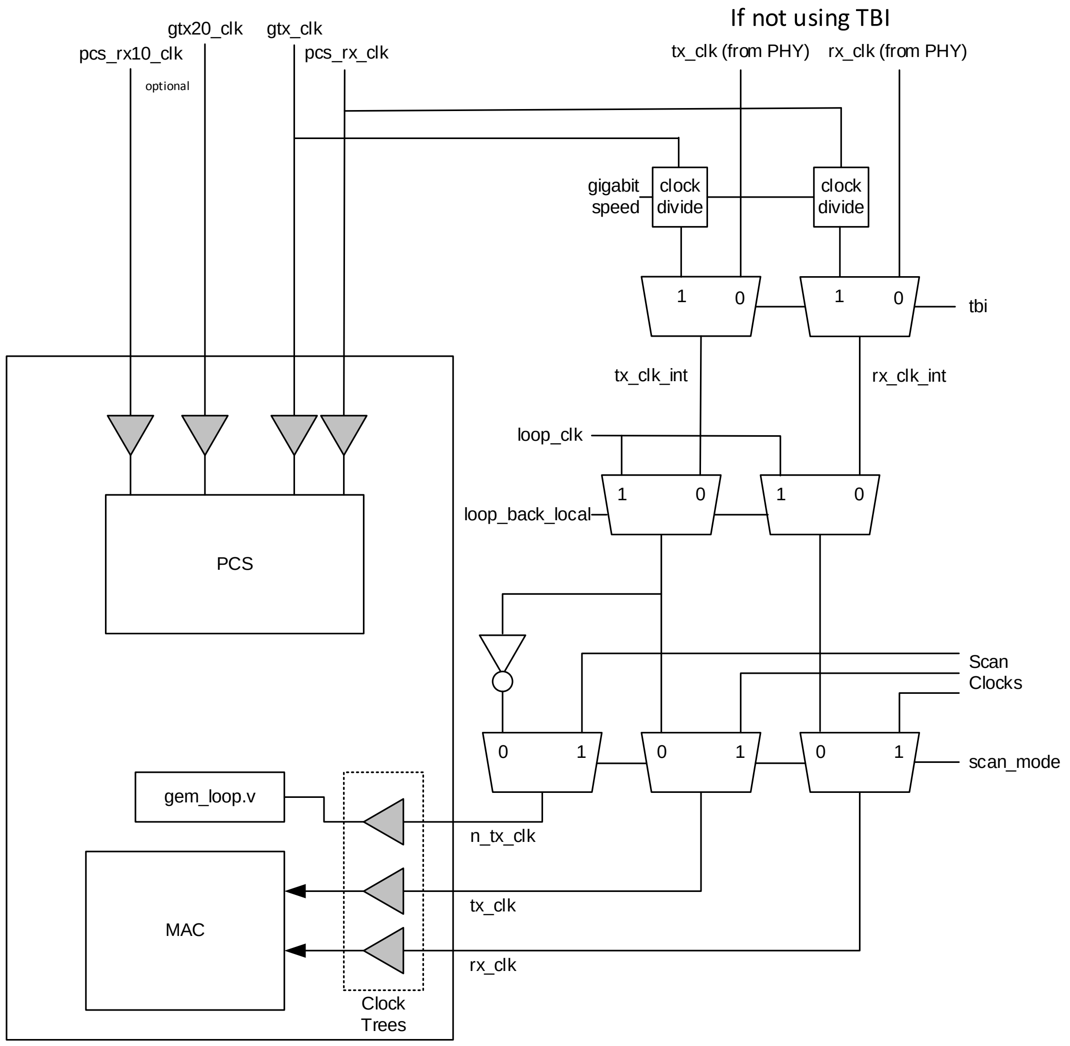
In the figure below tbi, gigabit and speed reflect the status of bits 11, 10 and 0 in the network configuration register and are output at the top level as speed_mode bits 2, 1 and 0.

```
  // speed_mode bits- 2 1 0            function
  //  -----------------------------------------------
  //        1 1 x      1000 Mbits/s using TBI interface
  //        0 1 x      1000 Mbits/s using GMII interface
  //        0 0 1       100 Mbits/s using MII interface
  //        0 0 0        10 Mbits/s using MII interface
  //        1 0 1       100 Mbits/s using SGMII interface
  //        1 0 0        10 Mbits/s using SGMII interface
```

The diagram below shows a typical clock tree for configurations using a legacy TBI interface with rbc0 and rbc1 clocks.



The following diagram shows the clock tree for using 10 and 20-bit PCS configurations:

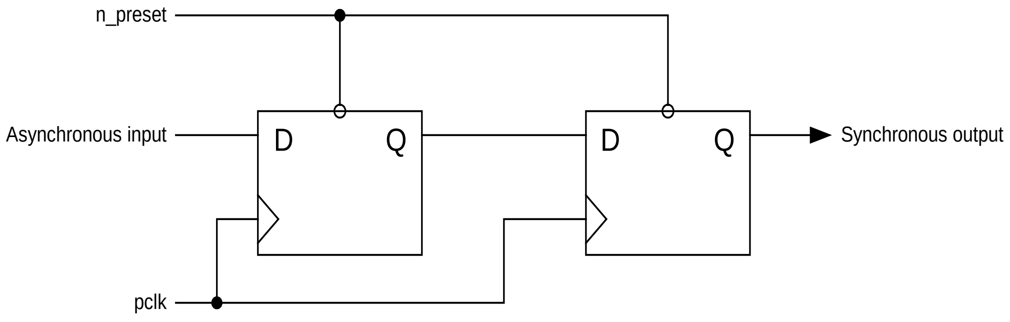


Resets

Each clock domain requires an active low reset to be provided. These resets are used directly to reset the synchronous parts of the design. The reset signals may be set low asynchronously with respect to the clock, but must be set high synchronously.

Metastability

To address the problem of metastability, all critical signals passing between clock domains are synchronised by passing through at least two flip-flops. Because the clocks can all be asynchronous with respect to each other, each data transfer across a clock domain is implemented using handshaking logic.



Synthesis Constraints

Example synthesis scripts are provided with the design to demonstrate how to constrain the clocks and I/O of the GEM. These are located in the scripts/rc subdirectory.

Signals passing between clock domains are controlled by handshaking logic. Even though there is no combinatorial logic between two flip-flops in different clock domains, this handshaking logic requires that the maximum delay path between the two clock domains is constrained to one clock period of the destination clock.

The receive path interface between the `gem_pcs` and `gem_rx` passes data between `rbc1` or pcs_rx_clk and `rx_clk` clocks. In gigabit TBI mode, the `rx_clk` should be sourced from `rbc1 or pcs_rx_clk` which requires that these two clock trees are balanced with each other.

Similarly the transmit path interface between the `gem_pcs` and `gem_tx` passes data between `gtx_clk` and `tx_clk` clocks. In gigabit TBI mode, the `tx_clk` should be sourced from `gtx_clk` which requires that these two clock trees are balanced with each other.

Combinatorial Paths

When configured for RGMII operation, there are two combinatorial paths, constrained by the `set_max_delay` constraint. These are from `rgmii_tx_clk_sig` and ending at both `rgmii_txd` and `rgmii_tx_ctl`;

```
set_max_delay 6 -from [get_ports rgmii_tx_clk_sig] -to [get_ports rgmii_tx_ctl]
set_max_delay 6 -from [get_ports rgmii_tx_clk_sig] -to [get_ports rgmii_txd]
```

Library Requirements

This design has no special cell requirements.

Please follow the guidelines of your library supplier when synthesising to make correct use of scan FF, clock distribution or similar specialised cells.

Verification & Test Application

Testbench structure

There is one supplied test-bench in the `func_ver/hvm` directory. It has been developed around the RTL description, and has been used with a synthesised net-list (not provided). Numerous tests are provided with the test bench.

Simulation must be performed in the `work` directory, which must contain a `results` and a `files` subdirectory. These are created automatically by the delivered scripts.

Test scenarios are stored in the `func_ver/hvm/tests/<configuration>` directories; these are well documented and can be used as a starting point for development of further test scenarios. These test scenarios are translated using a Perl program called `trans.pl`, stored in the `func_ver/hvm/tb` directory. The output is test stimulus and comparison data stored in the `work/files` directory.

Simulation scripts are supplied in the `func_ver/hvm/scripts` directory; these are configured to use the Cadence Incisive NC Verilog Simulator tool.

To simulate the test scenarios, the RTL and testbench code must first be compiled, this is accomplished using the `compile.pl` script held in the `scripts` directory and again assumes that Cadence Incisive NC Verilog Simulator has been properly installed.

For example from the `work` directory use:

`../scripts/compile.pl –cf <configuration>`

Once the RTL and testbench code has been compiled, the `simulate.pl` script should be called. To run all the tests, call the `simulate.pl` script with the option `-all`, otherwise use `-run [test case]`.

For example from the `work` directory use:
`../scripts/simulate.pl -run basic_rx`

Which will run the basic_rx test from the ../sim/stand_alone/tests_<configuration> directory. For further command switch options, the `-help` command switch is available. For example there are command switch options to compile the design for the different supplied configurations using `–cf`. Configurations include compiling for the packet buffer and for compiling with no DMA block.

Note that all the `scripts` are designed to be run from the `work` directory and execute with `../scripts/[script name]`. If the `work` directory is not the current directory, an error message will be displayed.

After a test scenario has been simulated a brief pass or fail status is written to the `results` directory. This status is written to a file with the same name as the test scenario with an extension of `.res`. A summary of previous simulation results can be returned by executing the Unix command `cat results/*` from the `work` directory. Additionally the simulator log file is copied to the `func_ver/hvm/reports/<configuration>` directory.

If a simulator other than `ncsim` is used take the following steps:

1. Compile and elaborate the RTL and testbench. The RTL source files are listed in the file `hdl/hdl_src/<design>.f` and the testbench source files are listed in the file s `func_ver/hvm/tb/tb_<design>.f`. The top level module for elaboration is called `tb_gem`.
2. Translate the test scenario using the `trans.pl` Perl program.
3. Invoke the simulator on the elaborated `tb_gem` module. The test bench will automatically stop at the end of the test scenario.

CPU Based Testing

No CPU based tests are supplied with this release of the IP.

Synthesizing the Design

Scripts are provided to demonstrate synthesising the design using Cadence RTL Compiler and SDC constraints for each of the example configurations are also provided.

In order to run synthesis, use the following steps:

- setenv STORK_DELIVERY 1
- cd synth
- mkdir work
- cd work
- ln –s ../../hdl_qc/cfg/<config>/setup_project.csh .
- ln –s ../../hdl_qc/cfg/<config>/project.tcl .
- ln –s ../../hdl/hdl_src/gem_gxl_defs_<config>.v ./gem_gxl_defs.v
- ../scripts/run_phys.csh –synth_rc

The scripts rely on the correct library information being set up. Refer to the following file and update accordingly:

```
synth/scripts/tech_lib_example.tcl
```